Welcome to the 33rd International Conference on Software Engineering and Knowledge Engineering (SEKE), in KSIR Virtual Conference Center, Pittsburgh, PA, USA. In the last 30 years, SEKE has established itself as a major international forum to foster, among academia, industry, and government agencies, discussion and exchange of ideas, research results and experience in software engineering and knowledge engineering. The SEKE community has grown to become a very important and influential source of ideas and innovations on the interplays between software engineering and knowledge engineering, and its impact on the knowledge economy has been felt worldwide. On behalf of the Program Committee, it is my great pleasure to invite you to participate in the technical program of SEKE.

This year, we received 196 submissions. Through a rigorous review process where a majority of the submitted papers received three reviews, and the rest with two reviews, we were able to select 67 full papers for the general conference (34 percent), 45 short papers (23 percent) and 84 rejects (43 percent). In addition, there are 5 demos selected from 7 demo submissions. SEKE 2021 Technical Program consists of four invited talk (keynote and plenary talk) sessions, 11 paper presentation sessions and one demo sessions. We greatly appreciate the committee members and authors of accepted papers in professional roles to serve as the chairs of the 16 sessions.

The high quality of the SEKE 2021 technical program would not have been possible without the tireless effort and hard work of many individuals. First of all, we would like to express our sincere appreciation to all the authors whose technical contributions have made the final technical program possible. We are very grateful to all the Program Committee members whose expertise and dedication made our responsibility that much easier. Our gratitude also goes to the keynote speakers who graciously agreed to share their insight on important research issues, to the conference organizing committee members for their superb work, and to the external reviewers for their contribution.

Personally, we owe a debt of gratitude to a number of people whose help and support with the technical program and the conference organization are unfailing and indispensable. We are deeply indebted to Dr. S. K. Chang, Chair of the Steering Committee, for his constant guidance and support that are essential to pull off SEKE 2021. Our heartfelt appreciation goes to Dr. Raúl García Castro, Polytechnic University of Madrid, Spain, the Conference Chair, for his help and experience. We also thank Dr. Jeff Offutt, Dr. Iaakov Exman, Dr. Adrian Riesco, and Dr. Enrique Alba for their excellent keynotes.

We would like also to express our great appreciation to all of the conference organization committee members, including the Publicity Chair, Rong Peng, Wuhan University, China and Publicity Co-Chairs Michael Bosu, New Zealand; Patrick Cook, Texas Tech University, USA; and Carlos Eduardo Pantoja, Federal Center for Technological Education, Brazil. Moreover, we would like to appreciate and recognize our Conference Liaisons in different regions for their important contributions. They are: Asia Liaison – Hironori Washizaki, Waseda University, Japan; Europe Liaison – Raúl Garcia Castro, Polytechnic University of Madrid, Spain; and India Liaison - Swapan Bhattacharya, National Institute of Technology Karnataka, Surathakl.

Last but certainly not the least, we must acknowledge the important contributions that the KSI staff members have made. Their timely and dependable support and assistance throughout the entire process have been truly remarkable. Finally, we wish you have productive discussion, great networking and effective virtual presentation to participate in SEKE 2021.

Kazuhiro Ogata, JAIST, Japan, Program Committee Chair
Lan Lin, Ball State University, USA, Program Committee Co-Chair
SEKE 2021

The 33rd International Conference on
Software Engineering &
Knowledge Engineering

July 1 – 10, 2021

KSIR Virtual Conference Center, Pittsburgh, USA

Conference Organization

CONFERENCE CHAIR
Raul Garcia Castro, Universidad Politecnica de Madrid, Spain; Conference Chair
Huiqun Yu, East China University of Science and Technology, China; Conference Co-Chair

PROGRAM COMMITTEE CHAIR AND CO-CHAIR
Kazuhiro Ogata, JAIST, Japan; PC Chair
Lan Lin, Ball State University, USA; PC Co-Chair

STEERING COMMITTEE CHAIR
Shi-Kuo Chang, University of Pittsburgh, USA

STEERING COMMITTEE
Vic Basili, University of Maryland, USA
Bruce Buchanan, University of Pittsburgh, USA
C. V. Ramamoorthy, University of California, Berkeley, USA

ADVISORY COMMITTEE
Jerry Gao, San Jose State University, USA
Swapna Gokhale, University of Connecticut, USA
Xudong He, Florida International University, USA
Natalia Juristo, Universidad Politecnica de Madrid, Spain
Taghi Khoshgoftaar, Florida Atlantic University, USA
Guenther Ruhe, University of Calgary, Canada
Masoud Sadjadi, Florida International University, USA
Du Zhang, California State University, USA

PROGRAM COMMITTEE

Silvia Teresita Acuna, Universidad Autonoma de Madrid, Spain
Shadi Alawneh, Oakland University, USA
Hyggo Almeida, Federal University of Campina Grande, Brazil
Dionyssios Athanasopoulos, Queen's University of Belfast, United Kingdom
Doo-Hwan Bae, Korea Advanced Institute of Science and Technology, Korea
Kyungmin Bae, Pohang University of Science and Technology, Korea
Kulaneswaran Banujan, Sabaragamuwa University of Sri Lanka, Sri Lanka
Vita Barletta, University of Bari, Italy
Ateet Bhalla, Consultant, India
Swapan Bhattacharya, Jadavpur University, India
Tanmay Bhowmik, Mississippi State University, USA
André Borges, Federal University of Technology – Parana (UTFPR), Brazil (EEAS)
Michael Bosu, Centre for Information Technology, New Zealand
Ivo Bukovsky, Czech Technical University in Prague, Czech Republic
Guoray Cai, Penn State University, USA
Juan Cano-DeBenito, Universidad Politécnica de Madrid, Spain
Rafael Cardoso, University of Liverpool, United Kingdom (EEAS)
John Castro, Universidad de Atacama, Chile
Samuel Chen, Brookhaven National Laboratory, USA
Xiang Chen, Nantong University, China
Wen-Hui Chen, National Taipei University of Technology, Taiwan
Lingwei Chen, Pennsylvania State University, USA
Xiangping Chen, Sun Yat-sen University, China (IPA)
Meiru Che, University of Texas at Austin, USA
Nacha Chondamrongkul, Mae Fah Luang University, Thailand
Lawrence Chung, University of Texas at Dallas, USA
Andrea Cimmino, Universidad Politecnica de Madrid, Spain
Patrick Cook, Texas Tech Universityy, USA
Fabio Costa, Universidade Federal de Goias, Brazil
Andrea DeLucia, University of Salerno, Italy (IPA)
Lin Deng, Towson University, USA
Wei Dong, National University of Defense Technology, China
Bowen Du, University of Warwick, United Kingdom
Weichang Du, University of New Brunswick, Canada
Abdulrahman Elfaki, University of Tabuk, Saudi Arabia
Iaakov Exman, Jerusalem College of Engineering, Israel
Onyeka Ezenwoye, Augusta University, USA
Yuan Fei, Shanghai Normal University, China
Maria Francesca Costabile, University of Bari, Italy
Fumiyo Fukumoto, University of Yamanashi, Japan (IPA)
Jianbo Gao, Peking University, China
Honghao Gao, ShangHai University, China
Raúl Garcia-Castro, Universidad Politècnica de Madrid, Spain
Ignacio García Rodriguez DeGuzman, University of Castilla–La Mancha, Spain
Swapna Gokhale, Univ. of Connecticut, USA
Wolfgang Golubski, Zwickau University of Applied Sciences, Germany
Desmond Greer, Queen's University Belfast, United Kingdom
Xudong He, Florida International University, USA
Dou Hu, National Computer System Engineering Research Institute, China
Sayem Imtiaz, Iowa State University, USA
Venkata Inukollu, Purdue University, USA
Bassey Isong, North-West University, South Africa
Clinton Jeffery, New Mexico Tech, USA
Shuyuan Jin, Sun Yat-sen University, China
Peiquan Jin, University of Science and Technology, China
Pankaj Kamthan, Concordia University, Canada
Taeghyun Kang, University of Central Missouri, USA
Ananya Kanjilal, B.P. Poddar Institute of Technology and Management, India
Ritu Kapur, IIT Ropar, India
Wiem Khlif, Miracle Laboratory, Tunisia
Taghi Khoshgoftaar, Florida Atlantic University, USA
Jun Kong, North Dakota State University, USA
Vinay Kulkarni, Tata Consultancy Services, India
Ashish Kumar Dwivedi, National Institute of Technology, India (IPA)
Olivier LeGoaer, University of Pau, France
Peng Liang, Wuhan University, China
Tong Li, Beijing University of Technology, China
Zengyang Li, Central China Normal University, China
Yingling Li, Chengdu University of Information Technology, China, (IPA)
Xin Li, Google Inc., USA
Yanhu Li, Nanjing University of Science and Technology, China
Lin Lin, Ball State University, USA
Bixin Li, Southeast University, China
Xiaodong Liu, Edinburgh Napier University, United Kingdom
Weidong Liu, Inner Mongolia University, China
Wanwei Liu, National University of Defense Technology, China
Yi Liu, University of Massachusetts Dartmouth, USA
Luanna Lopes Lobato, Federal University of Goias, Brazil
Jiawe Li, Zhejiang University of Technology, China (IPA)
Xinjun Mao, National University of Defense Technology, China
Beatriz Marin, Universidad Politecnica de Valencia, Spain, Chile
Riccardo Martoglia, University of Modena and Reggio Emilia, Italy
Baojun Ma, Shanghai International Studies University, China
Andre Menolli, Universidade Estadual do Norte do Parana (UENP), Brazil
Hid Milhem, University of Ottawa, Canada
Ran Mo, Central China Normal University, China
Eduardo Moraes, Federal Institute of Alagoas, Brazil
Hiro Yuki Nakagawa, Osaka University, Japan
Alex Norta, Tallinn University of Technology, Estonia
Kazuhiro Ogata, JAIST, Japan
Edson Oliveira Jr, State University of Maringa, Brazil
Carlos Pantoja, Federal Center for Technological Education (CEFET-RJ), Brazil
George Papadopoulos, University of Cyprus, Cyprus
Hyungbae Park, University of Central Missouri, USA
Rong Peng, Wuhan University, China
Oscar Mortagua Pereira, University of Aveiro, Portugal
Angelo Perkusich, Federal University of Campina Grande, Brazil
Chen Qian, Donghua University in Shanghai, China
Rick Rabiser, Johannes Kepler University, Austria
Claudia Raibulet, University of Milan, Italy
Rajeev Raje, IUPUI, USA
Marek Reformat, University of Alberta, Canada
Robert Reynolds, Wayne State University, USA
Daniel Rodriguez, Universidad de Alcalá, Spain
Azouzi Sameh, Laboratory RIADI-GDL, ENSI, Tunisia
Claudio Sant'Anna, Universidade Federal da Bahia, Brazil
Abdelhak-Djamel Seriai, University of Montpellier 2 for Sciences and Technology, France
Kaize Shi, Beijing Institute of Technology, China
Michael Shin, Texas Tech University, USA
Kazi Sultana, Montclair State University, USA
Xin Sun, Ball State University, USA
Meng Sun, Peking University, China
Yanchun Sun, Peking University, China
Zhenzhou Tian, Xi'an University of Posts and Telecommunications, China
Fadel Touré, UQTR, Canada
Mark Trakhtenbrot, Holon Institute of Technology, Israel
Christelle Urtado, LGI2P Ecole des Mines d'Ales, France
Dalton Valadares, IFPE Caruaru, Brazil
S. VasanthaPriyan, Sabaragamuwa University of Sri Lanka, Sri Lanka
Sylvain Vaughtier, Ecole des mines d'Ales, France
Gleifer VazAlves, Federal University of Technology - Parana (UTFPR), Brazil (EEAS)
Gennaro Vessio, University of Bari, Italy
Josã© Viterbo, Fluminense Federal University (UFF), Brazil (EEAS)
Jiaojiao Wang, China Communication University of Zhejiang, China (IPA)
Zhongjie Wang, Harbin Institute of Technology, China
Jian Wang, Wuhan University, China
Ye Wang, Zhejiang Gongshang University, China
Hironori Washizaki, Waseda University, Japan
Lingwei Wei, Chinese Academy of Sciences, China
Michael Weiss, Carleton University, Canada
Guido Wirtz, Bamberg University, Germany
Franz Wotawa, TU Graz, Austria
Ji Wu, Beihang University, China
Peng Wu, Institute of Software, Chinese Academy of Sciences, China
Xi Wu, The University of Sydney, Australia
Lai Xu, Bournemouth University, UK
Haiping Xu, University of Massachusetts Dartmouth, USA
Weifeng Xu, University of Baltimore, USA
Koji Yamamoto, Fujitsu Laboratories Ltd., Japan
Guowei Yang, Texas State University, USA
Huiqun Yu, East China University of Science and Technology, China
Jiang Yue, Fujian Normal University, China
Dongjin Yu, Hangzhou Dianzi University, China
Fiorella Zampetti, University of Sannio, Italy
Pengcheng Zhang, Hohai University, China
Du Zhang, Macau University of Science and Technology, China
Yong Zhang, Tsinghua University, China
Yongxin Zhao, East China Normal University, China
Yongjie Zheng, California State University at San Marcos, USA
Nianjun Zhou, IBM, USA
Huibiao Zhu, East China Normal University, China
Hongming Zhu, Tongji University, China
Eugenio Zimeo, University of Sannio, Italy

BEST DEMO AWARD COMMITTEE
Nazlie Shahmir, Canadian Pacific Railway, Canada, Chair
Tobias Hecking, German Aerospace Center, Germany
Tom Hill, Fellows Consulting Group, USA

PUBLICITY CHAIR AND CO-CHAIRS
Rong Peng, Wuhan University, China; Publicity Chair
Michael Bosu, New Zealand; Publicity Co-Chair
ASIA LIAISON
Hironori Washizaki, Waseda University, Japan

EUROPE LIAISON
Raul Garcia Castro, Universidad Politecnica de Madrid, Spain

INDIA LIAISON
Swapan Bhattacharya, National Institute of Technology Karnataka, Surathakl, India
Keynote: "Software Engineers Don't Care about Models, but They Want Automated Tests" by Dr. Jeff Offutt, George Mason University, USA

Plenary Talk: "Quantum Software Models: Instantiating Abstractions" by Dr. Iaakov Exman, Jerusalem College of Engineering, Israel

Keynote: "Declarative Debugging: Past, Present and Future" by Dr. Adrian Riesco, Universidad Complutense de Madrid, Spain

Plenary Talk: "How Can Metaheuristics Help Software Engineers" by Dr. Enrique Alba, University of Malaga, Spain

Session SAMS: Software Architecture Models and Systems

Refactoring Java Code to MapReduce Framework (S)

Junfeng Zhao and Rui Feng

1

ArchNet: A Concept-token based Approach for Determining Architectural Change Categories

Amit Kumar Mondal, Banani Roy, Sristy Sumana Nath and Kevin Schneider

7

Analyzing Open-Source Serverless Platforms: Characteristics and Performance (S)

Junfeng Li, Sameer Kulkarni, Ramakrishnan and Dan Li

15

I-CMOMMT: A multiagent approach for patrolling and observation of mobile targets with a continuous environment representation (S)

Jamy Chahal, Assia Belbachir and Amal El Fallah Seghrouchni

21

NVMSorting: Efficient Sorting on Non-Volatile Memory

Zhaole Chu, Yongping Luo, Peiquan Jin and Shouhong Wan

25

HHML: A Hierarchical Hybrid Modeling Language for Mode-based Periodic Controllers

Zhiming Hu, Zheng Wang, Hongjian Jiang, Yuyuan Zhang and Yongxin Zhao

31

Patterns for Reuse in Production Systems Engineering

Kristof Meixner, Arndt Lüder, Jan Herzog, Dietmar Winkler and Stefan Biffl

37

Software Design Pattern Analysis for Micro-Service Architecture using Queuing Networks (S)

Shikuo Chang, Hanzhong Zheng and Justin Kramer

45

Correctness Arguments for an SDN MAC Learning Algorithm (S)

Liang Hao, Xin Sun and Lan Lin

51

Multiclass Classification of Four Types of UML Diagrams from Images Using Deep Learning

Sergei Shcherban, Peng Liang, Zengyang Li and Chen Yang

57
Session ESR: Ethic Security and Reliability

On Integrating Ethicality in User Stories (S) 63
  Pankaj Kamthan and Nazlie Shahmir

A Collaborative Forensic Framework for Detecting Advanced Persistent Threats 67
  Weifeng Xu, Jie Yan and Daryl Stone

An Empirical Study on the Impact of Class Overlap in Just-in-Time Software Defect Prediction (S) 75
  Minyang Yi, Guisheng Fan, Huiqun Yu and Xingguang Yang

FSSRE: Fusing Semantic Feature and Syntactic Dependencies Feature for threat intelligence Relation Extraction 79
  Xuren Wang, Mengbo Xiong, Famei He, Peian Yang, Binghua Song, Jun Jiang, Zhengwei Jiang and Zihan Xiong

AAMR: Automated Anomalous Microservice Ranking in Cloud-Native Environment 86
  Zekun Zhang, Bing Li, Jian Wang and Yongqiang Liu

Unsupervised Anomaly Detection Based on System Logs 92
  Hao Chen, Ruizhi Xiao and Shuyuan Jin

From Vulnerability Anti-Patterns to Secure Design Patterns (S) 98
  Alok Ratnaparkhi, Onyeka Ezenwoye and Yi Liu

Session FV: Formal Verification

Formal Modeling and Verification of ICN-IoT Middleware Architecture (S) 102
  Hongqin Zhang, Jiaqi Yin, Huibiao Zhu and Ningning Chen

AnB2Murphi: A Translator for Converting Alice&Bob Specifications to Murphi 108
  Yongxin Zhao, Hongjian Jiang, Jin Lv, Sijun Tan and Yongjian Li

Formal verification of multitask hybrid systems by the OTS/CafeOBJ method 114
  Masaki Nakamura, Kazutoshi Sakakibara, Yuki Okura and Kazuhiro Ogata

Formal verification of IFF & NSLPK authentication protocols with CiMPG (S) 120
  Thet Wai Mon, Shuho Fujii, Duong Dinh Tran and Kazuhiro Ogata

Formal verification of Anderson mutual exclusion protocol by introducing an auxiliary variable (S) 126
  Naoki Asae, Duong Dinh Tran and Kazuhiro Ogata

Using LSTM to Predict Tactics in Coq 132
  Xiaokun Luan, Xiyue Zhang and Meng Sun

Formal specification and model checking of a recoverable wait-free version of MCS 138
  Duong Dinh Tran, Kentaro Waki and Kazuhiro Ogata

Fine-Grained Neural Network Abstraction for Efficient Formal Verification 144
  Zhaosen Wen, Min Zhang and Weikai Miao

Modeling and Verification of CKB Consensus Protocol in UPPAAL (S) 150
  Yi-Chun Feng, Yuteng Lu and Meng Sun

Formalization and Verification of Dubbo Using CSP 154
  Zhiru Hou, Jiaqi Yin and Huibiao Zhu

Conv-Reluplex: A Verification Framework For Convolution Neural Networks (S) 160
  Jin Xu, Zishan Li, Miaomiao Zhang and Bowen Du

Tree Ensemble Property Verification from A Testing Perspective 166
What Do Developers Reply To? An Empirical Study of the Top Unmanned Aerial Vehicles (UAVs) Apps (S)  
  
  Fatma Outay, David Dampier and Haroon Malik

Quantifying Synergy between Software Projects using README Files Only (S)  
  
  Roxanne El Baff, Sivasurya Santhanam and Tobias Hecking

Evaluating a Tool for Creating Bug Report Assignment Recommenders (S)  
  
  Disha Devaiya, John Anvik, Meher Bheree and Farjana Yeasmin Omee

Session RPC: Requirements and Program Comprehension

A family of experiments for evaluating the usability of a collaborative modelling chatbot  
  
  Ranci Ren, John Castro and Silvia Acuña

Evaluation of Chatbots Usability Experimentation (S)  
  
  Ranci Ren, John Castro and Silvia Acuña

From word embeddings to text similarities for improved semantic clustering of functional requirements (S)  
  
  Takwa Kochbati, Shuai Li, Sébastien Gerard and Mraïdha Chokri

Investigating Process Algebra Models to Represent Structured Requirements for Time-sensitive CPS  
  
  Mathilde Arnaud, Boutheina Bannour, Arnault Lapitre and Guillaume Giraud

Risk Analysis for Collaborative Systems during Requirements Engineering (S)  
  
  Kirthy Kolluri, Robert Ahn, Tom Hill and Lawrence Chung

Requirements Formality Levels Analysis and Transformation of Formal Notations into Semi-formal and Informal Notations (S)  
  
  Aya Zaki Ismail, Mohamed Osama, Mohamed Abdelrazek, John Grundy and Amani Ibrahim

Session RC: Recommendation and Classification

Spatial-Temporal Forecast of the probability distribution of Oceanic Nino Index for various lead times  
  
  Jahnavi Jonnalagadda and Mahdi Hashemi

Recurrent Neural Graph Collaborative Filtering  
  
  Beichuan Zhang, Zhijiao Xiao and Shenghua Zhong

Deep Self-Attention for Sequential Recommendation (S)  
  
  Beichuan Zhang, Zhijiao Xiao and Shenghua Zhong

A recommender system to assist conceptual modeling with UML  
  
  Maxime Savary-Leblanc, Xavier Le Pallec and Sebastien Gerard

Evaluating Visual Explanation of Bug Report Assignment Recommendations (S)  
  
  Shayla Bhuyan and John Anvik

A Practical User Feedback Classifier for Software Quality Characteristics  
  
  Rubens dos Santos, Karina Villela, Diego Avila and Lucineia Heloisa Thom

Improved Multiple Part Algorithm (IMPA) to extract multiple solutions for RNA sequence classification problem  
  
  Naoual Guannoni, Faouzi Mhamdi and Mourad Elloumi

Attention Guided Filter for Jointly Extracting Entities and Classifying Relations  
  
  Shaoze Chen, Su Wang and Wenxin Hu
Multi-Label Classification of Parrott’s Emotions

Abhijit Mondal and Swapna Gokhale

Special Session SEKEEO: Software Engineering and Knowledge Engineering Education

On Conducting Tests in Software Engineering Courses during the COVID-19 Pandemic (S)

Pankaj Kamthan

Revisiting UML Class Relationship Recovery for Online Education (S)

Dionysis Athanasopoulos

Extracting Prerequisite Relations among Concepts from the Introduction of Online Courses (S)

Zesong Wang, Kui Xiao, Zeqing Qin and Shihui Wang

A Technical Capability Evaluation Model Based Concept and Prerequisite Relation in Computer Education (S)

Jiwen Luo, Tao Wang, Junsheng Chang and Xiaoting Guo

Session KO: Knowledge and Ontology

Triangle Counting by Adaptively Resampling over Evolving Graph Streams

Wei Xuan, Huawei Cao, Mingyu Yan, Zhimin Tang, Xiaochun Ye and Dongrui Fan

Influence Maximization with Consideration of PageRank Centrality and Propagation Probability

Qi Chen and Rong Yan

Grasping or Forgetting? MAKT: A Dynamic Model via Multi-head Self-Attention for Knowledge Tracing

Deming Sheng, Jingling Yuan and Xin Zhang

Deep Similarity Preserving and Attention-based Hashing for Cross-Modal Retrieval

Shubai Chen, Song Wu, Yu Chen and Yuan Yuan

An Efficient ROS Package Searching Approach Powered By Knowledge Graph

Long Chen, Xinjun Mao, Yinyuan Zhang, Shuo Yang and Shuo Wang

KatGCN: Knowledge-Aware Attention based Temporal Graph Convolutional Network for Multi-Event Prediction

Xin Song, Haiyang Wang, Kang Zeng, Yujia Liu and Bin Zhou

SolDetector: Detect Defects Based on Knowledge Graph of Solidity Smart Contract

Tianyuan Hu, Zhenyu Pan and Bixin Li

SMART: Towards Automated Mapping between Data Specifications

Safia Kalwar, Mersedeh Sadeghi, Alireza Javadian Sabet, Alexander Nemirovskiy and Matteo Giovanni Rossi

Incorporating Presuppositions of Competency Questions into Test-Driven Development of Ontologies (S)

Jedrzej Potoniec, Dawid Wiśniewski and Agnieszka Ławrynowicz

Session ML: Machine Learning

Ride Hailing Service Demand Forecast by Integrating Convolutional and Recurrent Neural Networks

Zinat Ara and Mahdi Hashemi

Transfer Learning-based City Similarity Measurement: A Case Study on Urban Hotel (S)
Ganghua Zhang and Xiaoping Che
A hierarchical RNN-based model for learning recommendation with session intent detection

Jinyang Liu, Chuantao Yin, Xiaoyan Zhang, Kunyang Wang and Hong Zhou
Intelligent Preprocessing Selection for Pavement Crack Detection based on Deep Reinforcement Learning

Yan Lin, Guosheng Xu, Guoai Xu, Siyi Li, Jie Deng and Jiankun Cao
Inspect Defect of Power Equipment via Deep Learning Method

Qi Zhang, Qingsong Cai and Jianhui Zheng
A method for generating various style Chinese fonts in the absence of training data (S)

Kang Shi and Tian-Ming Bu
Multi-Fusion with Attention Mechanism for 3D Object Detection

Ning Wang and Ping Sun
SADA: Improved Data Symbolization and Optimization Method on HAR from Microscopic Perspective

Men Huichao and Wang Botao
HARP Pro: Hierarchical Representation Learning based on global and local features for social networks

Wei Zhang, Jing Yang and Fanshu Shang
A Volume-Aware Positional Attention-Based Recurrent Neural Network for Stock Index Prediction

Xinpeng Yu and Dagang Li
Session STPTA: Software Testing Program Test and Analysis

DeepSCC: Source Code Classification Based on Fine-Tuned RoBERTa (S)

Guang Yang, Yanlin Zhou, Chi Yu and Xiang Chen
Multi-Granularity Code Smell Detection using Deep Learning Method based on Abstract Syntax Tree

Weiwei Xu and Xiaofang Zhang
FCEP: A Fast Concolic Execution for Reaching Software Patches

Meng Fan, Wenzhi Wang, Aimin Yu and Dan Meng
Leveraging Compiler Optimization for Code Clone Detection

Shirish Singh, Harshit Singhal and Bharavi Mishra
Which Factors Affect Q-Learning-based Automated Android Testing? – A Study Focusing on Algorithm, Learning Target, and Reward Function –

Yuki Moriguchi and Shingo Takada
MACA: A Residual Network with Multi-Attention and Core Attributes for Code Search (S)

Lian Gu, Zihui Wang, Jiaxin Liu, Yating Zhang, Dong Yang and Wei Dong
Dynamically Detecting Invariants for Automatic Testing PLC Programs (S)

Zeyu Lu, Xia Mao, Yanhong Huang, Jianqi Shi and Yang Yang
Optimal Conjunctive Normal Form Encoding for Symbolic Execution

Weiyu Pan, Ziqi Shuai, Ke Ma and Luyao Liu
Graph queries for analyzing the coverage of requirements by test cases

Shingo Ariwaka, Hiroyuki Nakagawa and Tatsuhiro Tsuchiya
Using the Normalized Levenshtein Distance to Analyze Relationship between Faults and Local Variables with Confusing Names: A further Investigation (S)  
Carmine Gravino, Alessandra Orsi and Michele Risi  

A Framework for Mutation Testing of Machine Learning Systems (S)  
Raju Singh and Mukesh Kumar Rohil  

A Case Study of Testing an Image Recognition Application (S)  
Chuanqi Tao, Dongyu Cao, Hongjing Guo and Jerry Gao  

Session NLTPDM: Natural Language and Text Processing and Data Mining  

A Novel Text Classification Approach based on Meta-path Similarities and Graph Neural Networks  
Huan Wang, Jiang Li, Qing Zhou and Liang Ge  

Model-Agnostic Local Explanations with Genetic Algorithms for Text Classification  
Qingfeng Du and Jincheng Xu  

Towards a Better Understanding of Gradient-Based Explanatory Methods in NLP  
Qingfeng Du and Jincheng Xu  

Relation Extraction Model Based on Keywords Attention (S)  
Yu Chen, Jianxia Chen, Chang Liu and Qi Liu  

Chinese Sentence Semantic Matching With Multi-Granularity Based on Siamese Neural Network  
Xuan Wen, Jianxia Chen, Yu Chen and Shirui Sheng  

Complementary Representation of ALBERT for Text Summarization  
Wenying Guo, Bin Wu, Bai Wang, Lianwei Li, Junwei Sun and Maham Nazir  

BEHIND: a 4W-oriented Method for Event Detection from Twitter  
Kang Zeng, Yujia Liu, Xin Song and Bin Zhou  

ATFE: A Two-dimensional Feature Encoding-based Sentence-level Attention Model for Distant Supervised Relation Extraction  
Shiyang Li and Qianqian Ren  

Extracting information from driving data using k-means clustering (S)  
Nour Chetouane, Lorenz Klampfl and Franz Wotawa  

RoBF: An Auto-Tuning Bloom Filter for Mixed Queries on LSM-Tree  
Ruicheng Liu, Peiquan Jin, Shouhong Wan and Bei Hua  

Using Surrounding Text of Formula towards More Accurate Mathematical Information Retrieval  
Cheng Chen, Yifan Dai, Yuqi Shen, Jinfang Cai and Liangyu Chen  

Automatic Comprehension of Geometry Problems using AMR Parser (S)  
Anca-Elena Iordan  

Special Session DEMO: Technical Demos  

AutoCom: Automatic Comment Generation for C Code (D)  
Zhikang Tian and Yuekang Li  

Water-Wheel: Real-Time Storage with High Throughput and Scalability for Big Data Streams (D)  
Yanqi Lv, Ruicheng Liu and Peiquan Jin  

MT4ImgRec: A Metamorphic Testing Tool for Image Recognition Software (D)  
Dongyu Cao, Hongjing Guo and Chuanqi Tao
Post2Event: Extracting Key Events from Microblogs (D) 637

Chongwei Wang, Xujian Zhao, Peiquan Jin, Hui Zhang, Chunming Yang and Bo Li

CASTR: Assisting Bug Report Assignment Recommender Creation (D) 639

Disha Thakarshibhai Devaiya, John Anvik, Farjana Yeasmin Ome and Meher Bheree

Notes: (S) denotes a short paper. (D) denotes a demo description.
Keynote

Software Engineers Don't Care about Models, but They Want Automated Tests

Jeff Offutt, PhD
Professor of Software Engineering
Director, MS Software Engineering
Associate Chair of Graduate Studies, CS Department
George Mason University
USA

Abstract

Specification-based testing has roots in the mid-1970s, when Huang suggested testing telecommunications software by covering nodes in FSMs. More criteria for generating tests were invented during the 1980s and 1990s. Practicing software engineers seldom used formal specifications, so scientists developed informal modeling languages, which led to model-based testing (MBT), which features abstract tests on the model and concrete tests on the implementation. Although these techniques have great value, academic researchers need to remember that, to practitioners, models are simply a means to an end—a way to create high quality software. To focus our research on realistic industry problems, we need focus on the end result of high quality, well tested software, and use as much automation as possible to reduce the human cost of creating models, creating tests, and running tests. Software engineering research is at its best when it helps real software engineers create better software, cheaper.

About the Speaker

Dr. Jeff Offutt is a Professor of Software Engineering at George Mason University. In his 30+ year career in academia, he has led numerous collaborative research projects with funding from government and industry; designed and taught courses and degree programs at all levels; mentored dozens of students and junior faculty; succeeded in leadership roles at university and internationally in research; and won awards that reflected teaching, research, and service. His research results are used widely, and his teaching innovations are copied by many university educators. He has received several awards, including the Outstanding Faculty Award from the State Council of Higher Education for Virginia in 2019, GMU's Teaching Excellence Award in 2013, and Faculty of the Year award from GMU's alumni association for 2020. Offutt earned the PhD in Computer Science from Georgia Tech in 1988. He is on the web at cs.gmu.edu/~offutt/
Plenary Talk

Quantum Software Models: Instantiating Abstractions

Iaakov Exman
Software Engineering Department
The Jerusalem College of Engineering
Jerusalem, Israel

Abstract

Assuming that sooner or later quantum computing will materialize as a practical addition to classical computing, we foresee desirable features and design procedures for quantum and hybrid software systems. However, we claim that it is not reasonable to have totally distinct modular design procedures, one for classical software systems and another for quantum software systems. We argue in favor of a single unified design procedure for quantum, classical and hybrid software systems, reasoning that: (1) Classical software systems are classical limits of quantum systems – as usual in physics; (2) Software Systems instantiate Abstractions – any kind of abstractions, from numbers to higher-level conceptual systems; (3) Quantum and Classical Software have State/Operator duality – a quantum Density Matrix is both a state and a projection operator; a classical program is both a “readable code” state and a “runnable” operator on states. Our previous Linear Software Models enable formal linear algebraic procedures for modular design of classical software systems. Modularization has been performed by a spectral approach applied to matrix representations, e.g. the Laplacian of the software system. This approach clearly hints toward a single unified design procedure: (1) Linear Algebra – the basic objects of Linear Algebra also are the basic objects of quantum computation; (2) Laplacian Matrix – the Laplacian, a useful representation of classical software systems, is easily modified into a Density Matrix, a single unified representation of both classical and quantum software systems. We first point out the nature of problems that may occur while designing whole software systems involving quantum computation. Then we describe a proposal for a common design procedure starting point for both classical and quantum software systems, viz. Von Neumann’s quantum notion of Density Matrix. This proposal formulates modular design in terms of projection operators obtained from a design Density Matrix. We show, in the classical case, their equivalence to the Linear Software Models results obtained from the Laplacian matrix spectrum. The application in practice of the design procedure for classical, quantum and hybrid software systems is illustrated by case studies.

About the Speaker

Prof. Iaakov Exman is a faculty member of the Software Engineering Department at The Jerusalem College of Engineering, JCE, Azrieli, in Jerusalem, Israel. He got his M.Sc. degree from the Technion Institute of Technology, in Haifa, Israel, with a thesis on “Calculations with Gaussian Functions and a Model Hamiltonian”. His Ph.D. degree at the Hebrew University of Jerusalem was obtained with a thesis in the area of “Information Theory”. He has done post-doctoral research at Stanford University, CA, USA, working on “Computational Drug Discovery and Design”. After a long term industrial experience, in large industries in the AeroSpace area and in a small agile start-up in the Software Parallel Processing area, Prof. Exman returned to the academia to dedicate himself to Software Engineering. He has published a series of papers entitled “Linear Software Models” in international conferences, journals and book chapters. These papers focus on a formal and practical mathematical theory of software system design, based on algebraic structures, such as the Modularity Matrix, the Laplacian Matrix and conceptual Lattices. Recently, inspired by the Linear Software Models, a paper has been accepted for presentation in an international conference workshop, opening a new series on “Quantum Software Models”, which is the basis of the current plenary lecture. Iaakov Exman has collaborated with research groups in Spain, Germany, Sweden and Italy, and has scientific interactions with researchers in the USA. He is an editor in the area of Theory of Software Engineering in the Board of the International Journal of Software Engineering and Knowledge Engineering.

xviii
Keynote

Declarative Debugging: Past, Present, and Future

Adrián Riesco
Associate Professor
Department of Software systems and computation
Universidad Complutense de Madrid
Spain

Abstract

Declarative debugging, also known as algorithmic debugging, is a semi-automatic debugging technique that abstracts the execution order to focus on results. It proceeds by asking questions to an external oracle, usually the user, and has been successfully applied in programming languages such as Java, Haskell, and SQL. However, despite the nice properties of declarative debugging, it has not been widely used beyond Academy. In this talk we present the story and main features of declarative debugging, the projects currently in development, and the challenges it faces.

About the Speaker

Adrián Riesco is Associate Professor at Department of Software systems and computation from Universidad Complutense de Madrid, Spain (UCM). He received his PhD in Computer Science from UCM with his PhD thesis "Declarative Debugging and Heterogeneous Specification in Maude". His research interests are formal methods in rewriting logic and declarative debugging. He has published more than 60 papers on these topics, most of them in collaboration with national and international research centers. In particular, he has contributed in the field of declarative debugging for both imperative and declarative languages, in the theoretical and practical aspects, and in the integration with other paradigms, in particular with testing. Regarding teaching, he coordinates the Master studies in Computer Engineering and has taught several topics in the Mathematics and Computer Science faculties. More information is available at http://maude.sip.ucm.es/~adrian/
Plenary Talk

How Can Metaheuristics Help Software Engineers

Enrique Alba
Professor of Software Engineering
Director, MS Software Engineering
Associate Chair of Graduate Studies, CS Department
George Mason University
USA

Abstract

This presentation focuses on the potential benefits that metaheuristics (Genetic Algorithms, Ant Colonies, Particle Swarm, etc.) can bring to the field of Software Engineering (SE). For this to happen, we first need that a proper model of the SE problem is done in the form of an optimization, search or learning task. This is actually quite often the case in SE and other domains, thus allowing the utilization of powerful tools that can solve open problems in software testing, staff management for software projects, automatic tuning of communication protocols, model checking, next release problems, and a big amount of new challenges that can be now investigated thanks to the cross fertilization between these two domains. The talk will raise the main open questions in this new field as well as discuss on best practices, characterization, theory, and actual application of advanced search algorithms for software engineering.

About the Speaker

Prof. Enrique Alba had his degree in engineering and PhD in Computer Science in 1992 and 1999, respectively, by the University of Málaga (Spain). He works as a Full Professor in this university with varied teaching duties: data communications, distributed programing, software quality, and also evolutionary algorithms, bases for R+D+i and smart cities, both at graduate and master/doctoral programs. Prof. Alba leads an international team of researchers in the field of complex optimization/learning with applications in smart cities, bioinformatics, software engineering, telecoms, and others. In addition to the organization of international events (ACM GECCO, IEEE IPDPS-NIDISC, IEEE MSWiM, IEEE DS-RT, smart-CT…) Prof. Alba has offered dozens postgraduate courses, more than 70 seminars in international institutions, and has directed many research projects (9 with national funds, 7 in Europe, and numerous bilateral actions). Also, Prof. Alba has directed 12 projects for innovation in companies (OPTIMI, Tartessos, ACERINOX, ARELANCE, TUO, INDRA, AOP, VATIA, EMERGIA, SECMOTIC, ArcelorMittal, ACTECO, CETEM, EUROSOTERRADOS) and has worked as invited professor at INRIA, Luxembourg, Ostrava, Japan, Argentina, Cuba, Uruguay, and Mexico. He is editor in several international journals and book series of Springer-Verlag and Wiley, as well as he often reviews articles for more than 30 impact journals. He is included in the list of most prolific DBLP authors, and has published 130 articles in journals indexed by ISI, 11 books, and hundreds of communications to scientific conferences. He is included in the top ten most relevant researchers in Informatics in Spain (fifth position in ISI), and is the most influent researcher of UMA in engineering (webometrics), with 14 awards to his professional activities. Pr. Alba’s H index is 62, with more than 18,000 cites to his work.
Refactoring Java Code to MapReduce Framework

Rui Feng  
Inner Mongolia University  
Hohhot, China  
15506597902@163.com

Junfeng Zhao*  
Inner Mongolia University  
Hohhot, China  
csjzf@imu.edu.cn

Abstract—Cloud computing has evolved into an infrastructure tool for scientific research and computing application. For many enterprises, it has become a trend to migrate their applications from the local to cloud. To leverage cloud computing infrastructure, some legacy code for special business process need be refactored to the programming models of cloud computing. The desired approach is to design an automatic tool for refactoring legacy code to the target code that can execute on cloud computing platform effectively. In this paper a new approach is proposed, which can automatically refactor Java sequential program into MapReduce paradigm. The approach works by first translating input code into functional representation, with loops succinctly encapsulated by fold operations. Then, guided by the transforming rules, the approach generates equivalent MapReduce programs. The rules use group-by operations to enable greater parallelism. Finally, a series of mapping rules is applied to map immediate code to the target code running on Spark. A new tool designed using this approach, JMRT, was evaluated using real world benchmarks. The experimental results show that the approach can generate the desired MapReduce program and the business execution efficiency can be improved.

Keywords—Automatic Refactoring; MapReduce; Fold Operation; Parallelism; Program Transformation

I. INTRODUCTION

In the past decade, MapReduce [1] has attracted interest as a parallel programming model, independent of difficulties of distributed computation [2]. Main-stream MapReduce frameworks equip average developers with the tools that can instantly transform them into distributed system developers [3,4,5]. Specifically, we can select the appropriate variation by looking at the type information of the λm function used by map. The tool provides developers with abstract data-parallel operators map and reduce that shield them from the complexity of distributed computing. Therefore, the use of cloud platforms is increasing rapidly. In addition to developing new applications directly on the cloud, more and more applications are being migrated from local servers to cloud servers.

There is a problem in the process, how to reduce the migration cost. To maximize the performance of cloud applications, the code migration method needs to transform their programming model from sequential to parallel. This transformation is the most difficult part of the code migration procedure because it requires analyzing data dependency and then refactoring the source code. Sometimes, it is impossible to transform legacy code due to data dependency and the costs. Therefore, it is necessary to find a way to automatically translate sequential code into executable code under the MapReduce programming model. DOI 10.18293/SEKE2021-059

In the current research, most of the work aim to refactor the code using traditional parallel transformation methods. A common method is to obtain the code data access pattern and determine the order of retrieving data values during the program runtime, and then use the data reordering method to complete code transformation [6]. However, this method is not suitable for the transformation of Java code to MapReduce programming model, and many extensions to this method cannot effectively solve this problem. In addition, there is work aimed at refactoring code using design code templates. Paper [7] classifies the source code according to business logic, and then proposes corresponding reconstruction rules for each type. The approach is limited to specific access patterns, without considering all the code scenarios, it is necessary to propose a more comprehensive method to achieve effective reconstruction of Java code. To sum up, the existing refactoring methods are not perfect, and the refactoring tools are not mature enough. It is necessary to propose a more comprehensive method to achieve effective reconstruction of Java code in cloud migration.

In this paper, a comprehensive automatic translation method for sequential code to MapReduce programming model is proposed, which can effectively implement Java code to MapReduce code refactoring. A tool that implements our method is generated, which can handle complex input programs.

Fig. 1 illustrates the design of our method, which translates sequential code into equivalent MapReduce programs. The method includes the following steps. The first step of the method is to translate the input program into a functional representation via GSA (Gated Single Assignment) form [8]. Although the functional form is semantically equivalent to the original imperative code, unfortunately exposes no parallelism. since a simple data-parallel program consists of at least a map followed by a reduce. To address these problems, the transforming rules are designed to govern where map functions can be introduced in a semantics-preserving manner. In more complex cases where loop iterations access overlapping locations, this method uses Spark's groupByKey operation to group operations by access, exposing more fine grained parallelism than the previous method. Final, a series of mapping rules are designed to map the executable MapReduce program to the target Spark platform.

The structure of the article is as follows. In the approach presented in this article, code under two different programming models needs to be transformed, an IR (intermediate representation) is needed to assist in the transformation. Therefore, the generation process of functional IR is described in the chapter 2. Since Java sequential code cannot be transformed directly to intermediate code, a representation that helps Java code to be transformed to IR is needed. In the first section of the second chapter, this form is briefly described; in
the second section, the algorithm for translating sequential code to this form is shown; in the third section, the functional IR used in this paper is described; and in the fourth section, the rules for generating intermediate code are described. Once the functional IR has been got, the next step is to generate the executable code under the MapReduce programming model. Therefore, in chapter 3, we first introduce the transformation rules that introduce parallelism into intermediate code, and then introduce the mapping rules that generate executable code on the target platform. The refactoring tools and experimental validation are presented in chapter 4.

II. GENERATING FUNCTIONAL IR

A. Introduction of GSA Form

Since the source code cannot be directly transformed to the IR, a form is needed to help the transformation. GSA form is a representation based on static single value assignment (SSA) [9,10], which is a representation generated by static code analysis [11, 12]. The GSA form assigns unique names to variables in the program and embeds the gated predicate information \( \phi \) function, so as to realize the analysis of program’s data flow and control dependency information. Specialized gating functions (\( \gamma \), \( \eta \) and \( \mu \)) are introduced in GSA to include the predicate of conditional branches. Different pseudo-functions replace the \( \phi \) functions at different confluence nodes in the program control flow graph.

A program in GSA form is essentially a functional program, which facilitates program transformation. The functional intermediate form is derived from program in GSA form. GSA form is the basis for converting imperative loops to a MapReduce style.

B. Construction of GSA Form

An algorithm of translating source code to GSA form is proposed in this paper. Two steps are required to translate source program to GSA form. The first step is to insert the special assignment statement \( \phi \) functions into certain places of the program and replace the \( \phi \) function with \( \mu \), \( \gamma \) and \( \eta \) function. In the second step, each reference to \( \nu \) in the program is replaced by a reference to one of the new names \( \nu \).

In order to transform sequential code to GSA form, the data structure CFG [13] and the dominance relation [14] between nodes exists in the CFG are used in this paper.

Dominance Frontiers are used to find where \( \phi \) functions are needed. The dominance frontier \( \text{DF}(X) \) of a CFG node \( X \) is the set of all CFG nodes \( Y \) such that \( X \) dominates a predecessor of \( Y \) but does not strictly dominate \( Y \).

The method for placing \( \phi \) functions is: Whenever node \( X \) contains a definition of some variable \( \nu \), any node in the dominance frontier of \( X \) needs insert a \( \phi \) function for \( \nu \). The following code shows how to insert \( \phi \) functions. The input of the algorithm is the CFG information of the source program. Then, the output of the algorithm is a CFG which contain the \( \phi \) function. Several data structures are used: \( P \) is an array that stores the CFG nodes being processed. Process is an array of flags, one for each CFG node. When node \( X \) is currently added to \( P \), the Process(\( X \)) is 1. DomFron is an array of flags, one for each node. When the \( \phi \) function about variable \( \nu \) has been inserted in node \( X \), DomFron(\( X \)) is 1.

```
Algorithm: Insert \( \phi \) functions.
1    for each \( \nu \) do // Assign each variable as follows
2        \( P \leftarrow 0 \)
3        DomFron \leftarrow 0
4    for each \( X \in S(\nu) \) do //Iterate the node that contains the
5        Process \leftarrow 0
6        \( P \leftarrow \{x\} \)
7        \( \text{Process}(Y) \leftarrow 1 \)
8        while [\( P \)=1] do
9            take (x,p) //Fetch node X from the collection of nodes being
10            //processed
11            for each \( Y \in \text{DF}(X) \) do //Iterates through the nodes in the
12                if DomFron(\( Y \))\( =0 \)
13                    then do
14                        add \( \phi \)-function for \( \nu \) to \( Y \) //Inserts the \( \phi \) function of
15                        the variable \( \nu \) in the node \( Y \)
16                        DomFron(\( Y \)) \leftarrow 1
17                    then do
18                        Process(\( Y \)) \leftarrow 1
19                        \( P \leftarrow \{y\} \)
20                end
21            end
22        end
23    end
24 end
```

Use a more precise control predicate function in the code of the GSA form. The use of three functions improves the analysis accuracy of variables, and ultimately improves the accuracy of parallelization. The third step uses the following rules to replace the \( \phi \) function in the program to complete the transformation of Java source code to GSA form.

\( \mu \) function: Replaces those \( \phi \) functions at the head of a loop. Each \( \mu \) function combines the index value initialized by the loop with the index value calculated in the loop body. In the first iteration, the \( \mu \) function returns the first argument \( i_0 \), which is the value assigned before entering the loop, otherwise, it returns the second argument \( i_1 \), which is the value from the previous iteration. It involves the change of loop index and places it in the loop header.

\( \gamma \) function: Replaces those \( \phi \) functions located at the confluence nodes that have no incoming back edges. The back edge is the edge from the descendants to the ancestors in DT. The \( \gamma \) function selects the value of a variable computed by the if statements, and the condition in if statements as the parameter.

\( \eta \) function: Replaces \( \phi \) functions at the nodes that contain loop exit edges as incoming edges. It selects the last value at the end of the loop. The function is placed where the loop exits. The
GSA form is derived from the CFG of the source program, so the semantic equivalence is verified.

The following code shows the construction of the GSA form.

S is a stack that stores every write operation of variable v, one for each variable. Count is an integer, one for each variable. The Count value of any v represents how many assignments to v have been processed. The algorithm uses a depth-first search method to access the Dominator Tree (DT), which contains information about variables and is directly generated by existing algorithms, and the search starts with Entry, where the entry value of v is represented by an empty assignment on the right. In DT, the children of a node X are all immediately dominated by X. After renaming each variable v to vi, the search will continue to other nodes. When the statement of each node on the variable v is processed, the stack of the current variable v is cleared. The words predecessor and successor refer to CFG. The words parent, child, ancestor, descendant refer to DT.

Algorithm: Construct GSA form.
1. Count ← 0
2. S is empty
3. call Visit(Entry) //Start the search with the entry node
4. Visit(X): //When you search for node X
5. for each assignment A in X do //Do the following for the assignment statement in the node
6. for each variable v in RHS(A)
7. rename v with vi where i = top(S(v))
8. end
9. for each variable v in LHS(A)
10. i ← Count value of v
11. rename v with vi in A
12. push(S, i)
13. the Count value of v ← i + 1
14. end
15. end
16. for each v in X do //Rename a variable in a function within a node
17. rename j-th variable v in φ with vi where
18. i = top(S(v))
19. end
20. for each Y ∈ children (X) do //Continue to call the children of the current node
21. call Visit(Y)
22. end

C. Functional IR

In current work, the intermediate representation is typed lambda calculus. Lambda calculus is the basic mathematical theory of functional programming language [15]. It can describe and analyze programming language and use λ expression to represent programming language. The current IR is a typed functional language based on lambda calculus. Typed lambda calculus assigns each item in the lambda calculus a type, which can be int, string, etc. A simple example: if the variable x has type σ and there is an expression M, then λ x:σ.M defines a function that maps any x in σ to the value given by M. For translating an entire program block to λ expression, a reasonable typed grammar is let x = M in N, which means to constrain x to M within N. In other words, the value of let x = M in N is the value obtained by setting x to M in N. a[b] and a[b := c] are read and write access at index b of a map(array) a. Table 1 lists the set of operations in our IR. These expressions are well-known.

<table>
<thead>
<tr>
<th>IR operations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var a</td>
<td>b</td>
</tr>
</tbody>
</table>

"Figure 2 shows the data structure in the intermediate code and the functions operating on it. Most of these structures are well known, and they are described as needed in the article.

(data structures)
M[A]: multiset with values of type A
M[K, V]: map with keys of type K and values of type V
(map): (A → B) → (M[A] → M[B])
(map): (K → V) → (M[K] → M[V])
groupByKey: (A → K) → (M[A] → M[K])

Figure 2. Built-in data structures and functions.

D. Transform GSA Form to Functional IR

New transformation rules are proposed to transform GSA form code into an IR. Transformation rules transform loop and non-loop statements in code, respectively.

1) Non-loop Rules

The method transforms GSA form to the functional IR by applying the rules in Fig. 3. Simple rules are discussed first. Each non-GSA assignment statement is directly converted to the corresponding let ... in ... statement. Any branch instruction is skipped, left to be handled when reaching its associated node. The return instruction is replaced with the returned variable, which eventually sits at the innermost level of the let nest. Transform each γ instructions into a functional if statement whose condition comes from a branching instruction, and the branches be the arguments of the γ instruction. As the instructions are visited in topological order, the variables holding the result for each of the two branches are already available in scope.

let x = E < R → let x = E in R

a[x := y] → a[x := y]
return x → x

2) Loop Rules

The transformation of loops is essential to introduce parallelism, so it is very important to translate the loop structure. Fold operation is introduced to solve this problem. In functional programming, fold is a standard operator that encapsulates a pattern of function for processing recursive calls. Moreover, the fold operator is equipped with a proof principle called universality, which provides a mathematical principle for solving rule proofs. Fold has been introduced to ensure the correctness and availability of loop transformations.

let ... = fold ... (for(i = x, x; i < l, E[x_i = φ(x_i)]...)) →

let x = fold ... (for(i = x, x; i < l, E[x_i = φ(x_i)]...)) in range(i, l)

Figure 4. Loop Rules

The more complex rule translates loops to applications of the fold operator. In Fig.4, a loop is specified in terms of its φ
variables, which include the index variable i and other variables \(x_1, x_2\) updated in the loop body. These variables characterize all possible effects of the loop visible to subsequent code. For each \(\varphi\) variable \(x_\varphi\), we use \(x_\varphi'\) to refer to the value coming from outside the loop, and \(x_\varphi\) for the new value produced by the loop. The loop gets translated to a fold over the domain of values for the index variable, from i to l. The function \(f\) takes as arguments the current \(x_\varphi\) values, runs the body of the loop \(E\) once for those values, and returns the new \(x_\varphi\) values. The initial value for the fold is a tuple of the \(x_\varphi'\) values coming from outside the loop.

To obtain the fold operation over different types of input variables, the function \(E\) must first be obtained. The code is then transformed using the definition of the fold operation.

The definition of the functor of the following expression is used to construct the function \(E\). Functor \(E\) associated with each constructor \(e_i: (t_i,,\ldots, t_m) \rightarrow \mathcal{T}(a_i,,\ldots, a_p)\) is a \((p+1)\)-adic function (Where \(p\) is the number of universally quantified type variables in the left-hand side of \(\mathcal{T}\)’s type equation):

\[
E(f_1,,\ldots, f_p) = \lambda(x_1,,\ldots, x_{m_i}).(k[t]{x_i,,\ldots, k[t]{x_j})
\]

where the bound variable \(x_i\) has type, \(t_i\) and \(k[t]{x_j}\) represents a function that can be obtained by the following rules:

\[
k[t]{(a_i,,\ldots, a_p)} = f_p
\]

\[
k[t]{(a_i,,\ldots, a_p)} = \lambda x_{i_1}, x_{i_2}.(k[t]{x_{i_1}, k[t]{x_{i_2})}
\]

\[
[u \rightarrow v] = h \cdot k[u] \cdot k[v] \cdot h
\]

Now, it is possible to describe the fold operator for any loop with expression and functions. The fold function over \(T(a_1,,\ldots, a_p)\) is defined by the following equations.

\[
fold^d(f) = fold^d(\lambda T. f(x_0,,\ldots, x_n))
\]

where \(f = (f_1,,\ldots, f_n)\). Each \(f_i\) in \(f\) is a function.

The functional intermediate form is semantically equivalent to the original imperative code but unfortunately exposes no parallelism, since the loop operation is still sequential.

### III. Generating Executable MapReduce Code

#### A. Parallelize transformation rules

In the previous chapter, a functional IR with fold operations that is semantically equivalent to sequential code is generated. However, the current IR is still sequential and does not show any parallelism. Therefore, the parallelization transformation rule shown in Fig.6 is proposed to reveal the parallelism.

\[
<\text{extract map from fold}>\to\text{fold}_{\lambda T(x_0,,\ldots, x_n).E}f
\]

\[
(fold^d(\lambda T(x_0,,\ldots, x_n).f_{x_0}^E))\circ(map\ \lambda T_i(x_0,,\ldots, x_n).f_{x_0}^{E_i})
\]

\[
<\text{extract groupByKey}>\to\text{fold}_{\lambda r:v.E}f\circ(map\ \lambda v.C)
\]

Figure 5. Rules Revealing parallelism.

The transformation that reveals parallelism is the “extract map from fold” rule in Fig.5. It transforms a fold by extracting independent computations from the function \(f\) and transform it into a (parallelizable) map operation. An independent function must make no reference to an accumulator parameter. For example, \(\text{fold}^d(\lambda r:v.E)\) is transformed to \(\text{fold}^d(\lambda k:v.E)\), as \(f(v)\) is independent. After the transformation, the purely functional map can be easily parallelized.

The “extract map from fold” rule, shown in Fig.5, matches on any fold taking any type variables and functions. The fold operation \(E\) is split into the composition of functions \(f_0 \circ f_n\), such that \(f_n\) is independent of other “iterations” of the fold’s execution. If the fold is seen as a loop, \(f_n\) does not have any loop carried dependencies. \(f_n\) is pulled out into a map.

While the “extract map from fold” rule exposes significant parallelism, it cannot handle the situation when distinct loop iterations can update the same array location. MapReduce applications like word count mentioned earlier often work around such issues by using a shuffle operation to group inputs by some key and then process each group in parallel.

The transformation used for grouping by word is an application of the “extract groupByKey” rule shown in Fig. 5. The groupByKey operation clusters the elements of a collection of type \(M[A]\) according to the result of the function \(A\) to \(K\). It returns a map from keys \(K\) to lists \(M[A]\) of elements in the original collection that map to a specific key. The rule matches any fold with a body which is an update of a collection at an index \(E\).

The output code first groups the elements of the collection by the index expression \((\text{groupBy } \lambda v.E)\), and then it folds each of the groups using the update expression \(B\) from original body of the loop. groupByKey’s output is a Map from each distinct value of \(E\) to the corresponding subset of the input collection. The map operation’s parameters are \(K\), which bounds to the keys of the grouped collection), and \(I\) which contains subset of the input collection. The fold starts from the \(k\) value of \(r_1\), and folds \(I\) using the operation \(C\), which is original expression \(B\) with accesses to index \(E\) of the old reducer replaced with \(g\), the new parameter corresponding only to the \(k\)-index of \(r\).

#### B. Object Code Generation

When the program is translated, IR with MapReduce programming logic is generated. The mapping of the intermediate code to the target platform only involves the transformation of syntax in different programming languages and does not involve the conversion of semantics. Therefore, the corresponding mapping rules are constructed according to the syntax structure of the target platform and API calls. We list a subset of such mapping rules for the Spark RDD API [16].

\[
\text{map}(\text{Input,} T) \rightarrow \text{list(Pair)} = \text{Input.flatMapToPair(}[\lambda \lambda E])
\]

\[
\text{map}(\text{Input,} \lambda v: T) \rightarrow \text{list(U)} = \text{Input.flatMap(}[\lambda \lambda E])
\]

\[
\text{reduce}(\text{Input,} \lambda v: C) = \text{Input.reduce(}[\lambda \lambda E])
\]

An expression in the IR language is used as input, and the output is an equivalent expression in Spark. Since Spark provides multiple variants for the operators defined in our IR, such as a map, we can select the appropriate variation by looking at the type information of the \(\lambda m\) function used by the map. For example, if \(\lambda m\) returns a list of Pairs, we translate to JavaRDD.flatMapToPair.

### IV. Refactoring Tool and Experimental Evaluation

#### A. Refactoring Tool

Although there are many loops in legacy code, not all of them can be refactored in parallel based on the MapReduce model. Another work by our research group has proposed a way to identify parallelizable loops and annotate them with specific
parallel tags [17]. Based on the refactoring method proposed in this paper, a tool named JMRT supporting refactoring is designed. The tool obtains the code in the program according to the parallel mark, and then uses the components designed by the above method to refactor the code. The working process of the tool is shown in Fig. 6.

As seen in the picture, the components of JMRT comprise: receiving processor receives the sequential program that needs to be process; obtaining processor obtains the GSA representation of the source code; transforming processor translates the GSA representation to lambda representation; replacing processor replaces the loop with rules to generate executable parallel programs; and generating processor generates the executable MapReduce program on spark. In one example, any steps may be carried out in the order or the steps may be carried out in another order.

B. Experimental Verification and Result Analysis

1) Refactoring Experiment

In this section, a benchmark is used to test the feasibility of JMRT. Phoenix [18] is a standard MapReduce benchmark suite that provides both MapReduce and corresponding sequential implementations. It contains the main calculations from the application domain, such as enterprise computing (Word Count, Reverse Index, String Match), scientific computing (Matrix Multiply), artificial intelligence (KMeans, PCA, Linear Regression), and image processing (Histogram). Table 2 shows the number of loops and loop nests in the original programs, and whether the translation is successful or not.

<table>
<thead>
<tr>
<th>Program</th>
<th>Loop nests</th>
<th>Loops</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word Count</td>
<td>1</td>
<td>1</td>
<td>√</td>
</tr>
<tr>
<td>Histogram</td>
<td>1</td>
<td>1</td>
<td>√</td>
</tr>
<tr>
<td>String Match</td>
<td>1</td>
<td>1</td>
<td>√</td>
</tr>
<tr>
<td>Linear Regression</td>
<td>1</td>
<td>1</td>
<td>√</td>
</tr>
<tr>
<td>Matrix Product</td>
<td>3</td>
<td>2</td>
<td>√</td>
</tr>
<tr>
<td>PCA</td>
<td>2</td>
<td>5</td>
<td>√</td>
</tr>
<tr>
<td>KMeans</td>
<td>2</td>
<td>6</td>
<td>√</td>
</tr>
</tbody>
</table>

A concrete example is used to demonstrate how JMRT translates sequential code into MapReduce programs. Fig. 7 shows the sequential Java code, our starting point. The program iterates through a list of documents inputs, accumulating the word counts into the map.

```
1  static Map<String, Integer> wordCount(List<String> inputs) {
2      Map<String, Integer> map = new HashMap<>();
3      for (int i=0; i<inputs.size(); i++) {
4          String [] inputSplit = inputs.get(i).split(" ");
5              for (int j=0; j<inputSplit.length; j++) {
6                  String word = inputSplit[j];
7                      if (value == null)
8                          value = 0;
9                      else
10                          value += 1;
11                      map.put(word, value);
12              }
13      }
14      return map;
15  }
```

Then the program is translated into the functional IR. Thus, the code in Fig. 8 is converted to:

```
fold"((\(\lambda \ (k, v)\).  \\
\text{let} value = map[v] in \ \\
\text{if} value == null then 0 else value + 1) \\
\text{enddo})"(\(\text{inputs.Split}(" ")\))
```

Finally, the tool generates the code as follows:

```
\text{foldm}((\(\lambda \ (i, inputs)\). inputs.Split(" ")))
\text{map}((\(\lambda \ (key, value)\). (key, 1)))
\text{reduce}((\(\lambda \ (value0, value1)\). value0 + value1))
```

Each document is divided into multiple words, and then they are processed into (word, 1) pairs using the map function. The reduce function groups "1" values by their key, and then reduces the grouping by the add operation, thereby effectively counting the number of words. In this way, it reaches a form similar to the traditional MapReduce solution for the WordCount problem. The documents are split into words, which are then shuffled and the numbers of elements in each word package is counted. Finally, the corresponding Spark code is shown as follows.

```
val wordCount = inputs.split(inputs=>inputs.Split(" ")).(groupByKey\(\lambda \ \text{word} \ . \ \text{word}\))
```

Then the program is translated into the functional IR. Thus, the code in Fig. 8 is converted to:

```
fold"((\(\lambda \ (k, g \ w\ g+1)\).\(\text{groupByKey} \lambda \ \text{word} \ . \ \text{word}\))
```

Finally, the tool generates the code as follows:

```
\text{foldm}((\(\lambda \ (i, inputs)\). inputs.Split(" "))
\text{map}((\(\lambda \ (key, value)\). (key, 1)))
\text{reduce}((\(\lambda \ (value0, value1)\). value0 + value1))
```

Each document is divided into multiple words, and then they are processed into (word, 1) pairs using the map function. The reduce function groups "1" values by their key, and then reduces the grouping by the add operation, thereby effectively counting the number of words. In this way, it reaches a form similar to the traditional MapReduce solution for the WordCount problem. The documents are split into words, which are then shuffled and the numbers of elements in each word package is counted. Finally, the corresponding Spark code is shown as follows.

```
val wordCount = inputs.split(inputs=>inputs.Split(" "))
val pairs = wordCount.map(word=>\(\text{word}\))
val results = pairs.reduceByKey(\(\text{value0, value1}\) => value0 + value1)
```

2) Result Analysis

In terms of the experimental environment, the experiments were run on a quad-core Intel i7 at 2.6GHz with 16GB of RAM. One of the defining characteristics of translation results is performance, especially execution speed. We create a performance test experiment to test the performance of this translation method.

Fig. 9 compares the execution time of the sequential code and refactored code on Spark. The Spark translations generated for this benchmark performed 10.9× faster on average than the sequential versions. When the input data set is
small, the execution efficiency of sequential code is better than the executable code under the reconstructed MapReduce programming model. When the input data set is gradually enlarged, the advantages of the reconstructed code are gradually highlighted, and the corresponding execution efficiency is also continuously improved. This is because it takes a certain amount of time for each cluster to start and load data. When the input data set is small and there are many nodes, the time overhead brought by communication far exceeds the time advantage brought by parallel computing. Therefore, the execution efficiency of sequential code is better than that of the reconfigured code. However, the size of the input data set when the two types of code execution efficiency are demerged is not measured in this article, because it depends on many factors such as the network of the cloud platform.

This article describes a method for translating sequential Java code snippets into executable code under the MapReduce framework. Code snippets are transformed by defining a transformation rule in the method, which introduces the map and groupByKey functions to introduce parallelism. Our experiments show that JMRT can transform benchmarks in real-world applications. The generated code executes faster than the original code and is competitive with handwritten code.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (No.61962039) and Inner Mongolia Natural Science Foundation of China (No.2019MS0632).

REFERENCES

Architectural change classification is used to promote common approaches for addressing similar changes, produce appropriate design documentation for a release, construct a developer’s profile, form a balanced team, support code review, etc. However, automated architectural change classification techniques are in their infancy, perhaps due to the lack of a benchmark dataset and the need for extensive human involvement. To address these shortcomings, we present a benchmark dataset and a text classifier for determining the architectural change rationale from commit descriptions. First, we explored source code properties for change classification independent of project activity descriptions and found poor outcomes. Next, through extensive analysis, we identified the challenges of classifying architectural change from text and proposed a new classifier that uses concept tokens derived from the concept analysis of change samples. We also studied the sensitivity of change classification of various types of tokens present in commit messages. The experimental outcomes employing 10-fold and cross-project validation techniques with five popular open-source systems show that the F1 score of our proposed classifier is around 70%. The precision and recall are mostly consistent among all categories of change and more promising than competing methods for text classification.

I. INTRODUCTION

Software architecture is concerned with the partitioning of a software system into parts, with a specific set of relations among the parts [10]. A meaningful architectural document helps reduce the cognitive load and maintenance activities of the software development team [16]. Moreover, appropriate architectural formulation is becoming more critical to circumvent software bloat, scalability, and security backdoors [11]. However, elements of architecture can be changed [20] continuously as code components of a software system changes to support continuous development and maintenance [29] such as adding new features, restructuring the design models, and fixing flaws. Architectural change can affect many aspects of a software system and, for this, change analysis is a crucial task. Development team can group architectural changes leveraging change classification process based on the cause of the change, type of change, location of the change, the size of the code modification, and impact of change [40], [13]. For example, four major causes of architectural changes have been defined explicitly in the literature [40], [8], [29]: (i) perfective – adjusting new behaviour, (ii) preventive – prevent bad design, (iii) corrective – correct discovered problems, and (iv) adaptive – adapting to new platform.

Grouping causes of change is beneficial for post-release analyses, where design change activities are not explicitly annotated [8]. Change classification is also required for composing a developer’s profile, building a balanced team, and handling anomalies in the development process [21]. Furthermore, code review process involving architectural change is complex than local or atomic change [38], which is dependent on determining change type. Moreover, an automated technique can be employed to produce design documentation for every release recording types of structural changes happened and associated components [17]. Automated architectural change classification technique [40], [32] can be used to develop strategies for implementing a system change, support continuous architecture, augment DevOps and Model-Driven Engineering tools [6], [12], [11]. Existing active software projects (even if we consider a tiny portion of the 100 million repositories in GitHub [1]) could immediately benefit if a structural change classification technique is available to help develop an architectural versioning schema.

However, while architectural change can be identified from source code change, identifying the design decision, reason, and categories of changes requires analyzing the development team’s intention. The intention can be extracted from textual description of the developer’s tasks and discussions [6], [29]. Literature has focused on classifying typical software changes, architectural design concerns and design solutions [41], [28], [14]. Yet, supporting architectural change classification is still in its infancy [29], [11], perhaps, due to lack of benchmark data and requirements of laborious human analysis. Nevertheless, a few of the studies explored for both manual [8], [32] and semi-automated [29] techniques for classifying architectural changes. In these studies, a small collection of samples is being experimented where challenges are not identified properly, which leads to developing infeasible models. Besides, the traditional text classification techniques [7], [42] might not handle the scenario when keywords are present among multiple concepts within the description of a task.

To address the shortcomings, we design a benchmark data and propose a text classifier called ArchiNet for architectural change classification. In particular, we focus on the two research questions: RQ1: How can source code properties that are independent of the description of project activities
classify the rationale of architectural changes?, and RQ2: How can we improve text classification to predict the rationale of architectural changes leveraging commit descriptions?

To answer RQ1 and RQ2, we collect around 1,133 architectural change instances from 5K commits of five popular projects (shown in Table I). After extensive analysis of the created dataset, we have successfully identified the challenges of categorizing the architectural changes both from the source code and the texts. One of the challenges is that typical operations in the source code do not have a significant number of distinguishing patterns in various changes, and classification performance is not promising (F1 score is 33%). A major challenge in the commit description is that multiple concepts are presented, whereas only one or two concepts indicate the intention. Furthermore, many words are common for expressing the reasons for changes, such as keyword update is used to describe both perfective and adaptive changes. Such a phenomenon is not acute in many other text classification tasks [19]. All things considered, we propose a new technique for classifying the changes from the text where trained keywords from concept analysis of different changes play a crucial role. The training process of our proposed technique is different from the traditional NLP training process. For training, we first define the relevant concepts (contextual occurrence of words and tokens such as {update, API, version} indicate adaptive change more confidently) within each sample. Next, all tokens’ weights appeared within all the concepts for a relevant change validity. Section VIII discusses related studies and Section IX concludes our paper with future direction.

II. BACKGROUND

Architectural Change Instance: Studying typical changes from version control systems does not require a change detection strategy as it provides differences. However, architectural change detection [20], [24], even from the version control system (diff), is challenging. Some of the widely used change metrics are DSM [5], MoJo [39], MoJoFM [39], graph kernel structure [50], A2A [20], C2C [24] and include-symbol dependencies [24]. These metrics are calculated based on the following operations: adding components, removing components, replacing components, splitting components, merging components, relocating, module dependency graph, and usage dependency. We focus on intermediate-level architecture for collecting change samples and employ A2A and include-symbol dependency metrics for change detection. A2A considers component addition, removal and moves; include+ symbol dependency considers including/removing header file, program file, importing class, and importing interface. Causes for architectural changes are grouped as follows.

Adaptive (A) change: This change would be a reflection [40], [22], [23] of system portability, adapting to a new platform such as commit(1) in Fig. 1. Adaptive change also happen for imposing new organisational and governmental policies.

Corrective (C) change: A corrective change is the reactive modification of a software product performed after deployment to correct discovered problems [40]. Specifically, this change refers to defect repair, and the errors in specification, design and implementation.

Preventive (PV) change: Preventive change [37], [40] refers to actionable means to prevent, retard, or remediate code decay. In other meanings, preventive changes happen to improve file structure or to reduce dependencies between software modules and components may later impact quality attributes such as understandability, modifiability, and complexity.

Perfective (PF) change: Perfective changes are the most common and inherent in development activities. This change mainly focuses on adjusting new behaviour or requirements changes [37]. Also, these changes are aimed at improving processing efficiency and enhancing the performance of the software (such as commit(2) in Fig. 1) that is both functional and non-functional optimizations.

This classification is essential to deal with various challenges (discussed in the Introduction) since different types of change influence them in different ways. Among the change categories, preventive and corrective changes are directly related to major design debt management. A few of the change types in the two commits in Hadoop is shown in Fig. 1. Commit descriptions simply express their intentions. Commit(1) is an adaptive change in 2015 and commit(2) is a perfective change in 2018. It is noticeable from commit(2) that a dependency change between two components (htrace and hdfs) increases performance by reducing CPU usage, which is also an architectural change. Both of the changes happen almost a decade later of the first commit.
TABLE I: Candidate projects for our study (in inspection time).

<table>
<thead>
<tr>
<th>Project</th>
<th>All</th>
<th>A.</th>
<th>Domain</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>2303</td>
<td>266</td>
<td>Distributed Computing</td>
<td>github.com/hadoop</td>
</tr>
<tr>
<td>HibernateORM</td>
<td>9811</td>
<td>261</td>
<td>Object/Relational Mapping</td>
<td>github/hiber...hibernate-orm</td>
</tr>
<tr>
<td>LinuxTools</td>
<td>1959</td>
<td>255</td>
<td>C/C++ IDE for Linux</td>
<td>github/ci/ci-mixtools</td>
</tr>
<tr>
<td>Java/lombok</td>
<td>1437</td>
<td>136</td>
<td>Java bind for Appium tests</td>
<td>github/appium/java-client</td>
</tr>
<tr>
<td>JVM/Couchbase</td>
<td>914</td>
<td>205</td>
<td>JVM core for Couchbase Server</td>
<td>github/couchbase/couchbase-jvm-core</td>
</tr>
<tr>
<td>Total</td>
<td>45463</td>
<td>1133</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A: architectural changes in selected 1K commits; gt: github.com

release of Hadoop. Components of the htrace module are at the center of these two changes (commit(1) and commit(2)) although the second change occurred after three years of the occurrence of the first change.

III. DATASET PREPARATION

A few of the studies [8], [29] created datasets for architectural change classification from the development history. The recent dataset created by Mondal et al. [29] consists of 362 samples of four projects (26 of them are adaptive). This dataset might be insufficient for detecting some of the text classification challenges such as various concept tokens including code elements and framework name. Created dataset by Pixao et al. [32] contains architectural change only for new features and other categories are not annotated (recently they updated their dataset with fixing issues but not specifically annotated to four groups discussed widely in the literature [40], [8], [29]). Another dataset is constructed by Ding et al. [8] which is not publicly available (thanks to the authors for providing us 37 samples). Therefore, we prepare a new dataset containing a large collection of commits (shown in Table I).

1) Architectural Change Commits Filtering: We selected five open source projects that are widely experimented in literature for software change and architectural analysis [28], [20], [17], [32], ensuring a diversity of domains. We also ensure that the projects are in active development for at least several years. The selected projects are: Hadoop, Hibernate ORM, Linux Tools, Java Client, and Couchbase JVM Core have 45,463 commits which are infeasible to analyze manually. Since determining and categorising architectural change instances require huge human efforts, in our dataset creation process, we restrict primary selection of commit samples into 5K. We randomly choose 1K commits from each of the projects containing more than two words in the messages excluding stop words, non-alpha words (that having non-letters such as issue-110) along with the words having Change-Id: or Signed-off-by: and so on as shown in Fig. 1. We separate the architectural change samples from the primary collection (around 5K) if A2A and include + symbol dependency metrics are changed. However, as suggested by the literature [25], we do not consider system library usage from native (Java, Python) framework for dependency change. In this way, we get around 1133 samples (distribution of them is shown in Table I) as architecturally changed commits.

2) Architectural Change Category Annotation: In the next step of our study, we manually label those samples by two authors independently into one of the four categories described in the existing studies [29], [8]. There are ambiguities in some of the descriptions of four types of changes. We review most of the relevant papers referred by [40], [8], [29] for more explanation to resolve the ambiguity (details are discussed in Section V). Our manual annotation process has two iterations. In the first iteration, two of the authors having three years of average software industry experience, categorized the samples separately. In this step, we get many samples mismatched in annotation. In the second iteration, we recheck the mismatch samples and resolve the disagreements by discussion. Total number of samples in each of the annotated categories from the candidate projects is shown in Table II. The finalization of our dataset took one month of two person-hours, indicating that manual change analysis is expensive. In the next section, we investigate the automatic change classification challenges.

IV. CHANGE CLASSIFICATION CHALLENGES

For examining the challenges of classification, we divide the samples into two parts: training and test sets. As empirical study [18] suggests that 30% test samples are ideal for real data, we split around 70% of the architectural commits for training purposes and around 30% for testing purposes with random sampling. However, we could not extract meaningful concepts (Section IV-B2) for some of the samples due to lack of information, and skip those during the training and testing phases. Distribution of change types in the train and test sets are shown in Table II. Both the train and test sets contain the conflicted samples accordingly.

A. Classification from Source Code

First, we explore classification options leveraging source code operations. Yamauchi et al. [41] cluster the change commits based on source code modifications: identifiers, method name, and class name into as many groups dependent on component-requirement relations. Their technique cannot be used for a fixed number of classes. The clustering basically groups the commits into related components attached to an implemented functional requirement, not the reason for changes. Therefore, we explore a technique utilizing the distribution of change operations of the architectural components (static). We examine the abstract operations (O) occurred in the source code of a commit: import added or deleted, class file added or deleted, file or package rename, and function added or deleted as properties of change classification since they are universal and independent of project context.

Considering these properties, we design a classifier using $C_i(w_o)$ in (1) as described in Section V to evaluate how significant the prediction is using these operations as metrics and has the following outcome with 10 fold cross-validation. The best F1 score (among different combinations of the operation types) for perfective, preventive, corrective,
adaptive, and all combined are 0.33, 0.53, 0.08, 0.13, and 0.33 respectively. F1 for the corrective and adaptive classes are negligible. In summary, source code properties are not promising for architectural change classification; this answers our RQ1. In the next section, we explore existing change classification techniques from commits messages.

B. Change Classification from Text

1) Explored Models: Next, we examine the explored models of Mondal et al. [29], where the best model produces 39% F1 scores with our dataset. Following these approaches, we also develop a discriminating feature selection (DFS) model from the distribution of words in our training dataset. Similar to Mondal et al., our DFS model has many common keywords in the top list. Considering such overlapping of keywords, existing techniques based on the DFS model discussed in DPLSA [42], LLDA [34], and SemiLDA [9] predict more false positives since such a model also considers the words that might be irrelevant to the original intentions. With our new collection, the best DFS model produces 46% F1 score with precision 45.6% which is similar to the outcome of the best model in [29]. Our DFS model for the dataset in [29] produces an F1 score of 20% that is significantly lower than the previous model.

In summary, the DFS models are not promising and possibly biased to the project contexts. Therefore, we focus on a more advanced classifier identifying the challenges within the textual descriptions. We discuss classifiers from traditional machine learning and neural word embedding models in Section V.

2) Concept Analysis: As we have a large number of samples, we are able to identify the specific challenges within the message description. One of the significant challenges present in many commit messages is developers express more than one concept (contextual occurrence of words) for a single intention. An N-gram model might capture continuous sequences of n-words involved in such concepts within a sentence [35, 36]. However, in multiple iterations of our inspection, we find that concept words are scattered among multiple sentences in many commit descriptions. We also prioritize such scattered words while categorizing the commits. Traditional text classification techniques do not address this particular scenario (including the n-gram model). We also attempt to determine the dominating concepts from multiple concept tokens. Lets consider the corrective change message “adding more support for services down...” here adding support and down keywords will influence to predict a category by tf – idf [31], LLDA, SemiLDA, and DPLSA techniques. Unfortunately, adding and support keywords will measure more weight to other categories because they are present among the list of the top keywords. However, if we prioritize the down keyword as the dominating concept, it is more likely to be a corrective category. We annotated such keywords for the dominating categories.

In the list from Section V-B1 some dominating words (such as issue and leak) for this corrective category are hardly used for others. But, many samples contain negative words which are not meant faults, such as “...This changeset moves the responsibility of sending into the locators, which has two

<table>
<thead>
<tr>
<th>Base words</th>
<th>Not failure</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not</td>
<td>complex</td>
<td></td>
</tr>
<tr>
<td>Doesn't</td>
<td>need</td>
<td>work, release</td>
</tr>
<tr>
<td>Error</td>
<td>message</td>
<td>-- network, -- fix</td>
</tr>
<tr>
<td>Can't</td>
<td>change</td>
<td></td>
</tr>
</tbody>
</table>

Symbol ‘−’ indicates located before the base word.

TABLE III: Ambiguity of concepts appeared in description.

V. OUR PROPOSED CLASSIFIER: ARCHINET

From the empirical observations, it is evident that handling overlapping words among the descriptions is the key to develop a promising solution. We conjecture that no word should be in the distinguishing list to a single category. Instead of the logic of the previous techniques, we assign a strength of a word for each of the categories. For example, for the strength of the words presented in Table IV for different concepts, if the words add, support, down appear within a text description, then the total value for the category C1 is 0.52 + 0.38 + 0 = 0.90, and the total value for the category C2 is 0.03 + 0 + 1 = 1.03. As 1.03 > 0.90, the sample would be for the category C2. For simplicity, we explain with weight addition; more complex situations (with various token strengths) are handled with a probabilistic prediction technique described in Section V.

Therefore, this gives more importance to the co-occurrence of the words add and down, and such a solution might handle the described challenges in a promising way. In our solution, the crucial point is to get the concept tokens and their weights
distribution efficiently, and then predict a class confidently. We describe our proposed method in three steps.

<table>
<thead>
<tr>
<th>Word</th>
<th>Strength in $C_1$</th>
<th>Strength in $C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>0.52</td>
<td>0.03</td>
</tr>
<tr>
<td>support</td>
<td>0.38</td>
<td>0</td>
</tr>
<tr>
<td>down</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

1) Concepts Extraction: In this stage, we define and extract concepts from the commit messages of all the annotated samples that express the corresponding intention of a task (as discussed in Section IV-B2). Even, the top words (such as support) among the defined concepts contain many overlapping words. However, we have found some patterns in many samples for expressing different concepts when these terms are co-occurred with other tokens which are stop words, code elements, and API, library or framework name. Some of the examples are discussed in earlier sections. Before training, natural words are stemmed with PorterStemmer. In the next section, we discuss our training and weight distribution process from the extracted concept tokens.

2) Training Model Generation: In this phase, we train a model by assigning weights to the concept tokens using (1) from a set of preclassified commits into four change categories. This is motivated by the core idea of how the model for a word’s sentiment is generated (2). These weights represent the strengths while present within the concepts of the categories. The trained model produces a collection of unique concept tokens denoted by $S$ having weights $w_i$ to the classes $C_i$:

$$C_i(w_S) = \underset{t \in S}{\bigcup} w_i(t) \cup C_i(w_O), \quad S = \{t_w, t_p, t_s, t_a\}$$

(1)

$$w_i(t_w) = \frac{f(t_w)}{N_t}, \quad w_i(t_p) = \frac{F(t_p)}{N_t}, \quad w_i(t_s) = \frac{F(t_s)}{N_t}, \quad w_i(t_a) = \frac{F(t_a)}{N_t} \quad \text{where } O = \{t_o\}$$

(2)

Here, $f(t)$ frequency of a token $t$ within the concepts $S$ of all samples in a category $N_t$, and $C_i(w_O)$ is the weight of the tokens defined with source code change operation types ($O$) associated with $S, C_i \{PF, PV, C, A\}$. $w_i(t)$ can be calculated by adopting various metrics such as frequency value or $tfidf$. A concept $S$ consists of various types of tokens such as $t_w$ is natural words without stop words, $t_p$ is some specific stop words such as negation words, $t_s$ is some special code elements such as NullPointerException and LinkerError, $t_a$ is api, library or framework name, and $t_o$ is code operation types treated as tokens; each of these types has a collection of tokens. We utilized frequency normalized sum for calculating the probability value. We calculated the frequency values differently represented by bold $F$ in (2) and (3). $F(t_w)$ considers only inclusive stop words, $F(t_p)$ consider the issue related token parts (such as Exception and Error from the mentioned tokens) extracted from a code element using camel case parsing, $F(t_o)$ is calculated by converting all the api names into a unique token (“AA/BB” in our experiment), and $F(t_a)$ consider one or more instances of each token in $t_o$ as value 1 within a commit.

All the values in (1) and (2) are adjusted when new concepts are defined with new trained samples. Then, we employ a classifier from these trained weights.

3) Classification: During the classification phase, the generated models ($M$) from the training phase (in (1)) are used to evaluate the probability ($P_m(C)$) that a given class $C$ is associated with the commit $m$. Only the tokens identified in the concepts ($S$) from phase one and tokens as the source code operations ($O$) in Section IV-A are considered from commit message and code change. The classification score is then defined as follows.

$$P_m(C) = \frac{\sum_{t \in (S \cup O) \cap C} M(w(t))}{\sum_{t \in (S \cup O) \cap C} M(w(t))}$$

(4)

where the numerator is computed as the sum of the token weights ($w(t)$) of all types that are contained in $C$, and the denominator is the sum of the token weights for all types for all classes ($C_i$). The probabilistic classifier for a given commit $m$ will assign a higher score $P_m(C)$ to class $C$ that contains several strong tokens for concept $S$ and operation $O$. However, if the probabilities $P_m(C_i)$ are same for more than one class, ArchiNet considers the class which contains the highest weighted word.

VI. PERFORMANCE EVALUATION

Our created dataset contains both an architectural change set three times larger than that of [29] and a list of concept-words with strength. Our proposed classifier ArchiNet is designed to handle the overlapping words and includes various tokens discussed in Section IV-B2 within the change description. We compare the performance of ArchiNet based on recall (R) – quantitative correctness of retrieving relevant categories; precision (P) – the rate of accuracy among the predicted samples, and the F1 score – $2PR/(P + R)$ calculated from precision and recall. We have also compared with the published dataset and classifiers [29], [42]. The performance is also compared with other promising techniques in literature [14], [28], [36], [23] for text classification. These techniques include RCNN-LSTM the state-of-the-art Deep Neural Learning (DNL), Naive Bayes (NB), Bag-of-words (BoW) model, Decision Tree (DT), Random Forest (RF), DPLSA, LLDA, and SemiLDA. Our training model is significantly faster than RF and DNL, but we will not discuss time complexity since it is less critical if a model is built once for application. We evaluated the performance of ArchiNet in the following four phases.

1) Testing with the Golden Set: We train our proposed method (ArchiNet) and other methods with the training set. Train and test set partitioning is described in Section IV. Then, the classification performance is tested with the test set (from Table IV). comparison of the outcome is presented in Fig. 2. Please note that only methods having close performance are shown here. The most promising method in the baseline work by Mondal et al. is DPLSA, where discriminating keywords for the individual classes are used as features in a probabilistic model. The difference in the percentage of F1 score between
ArchNet and DPLSA for all classes is 24 points higher, while this difference is 35 points higher for the adaptive category. The F1 score of our model for the test data in [29] is 63% (shown in Table V), which is 18 points higher compared to their best model (45% gain in performance). We have employed a DNL-based text classifier [19, 36] with Google Tensorflow [3]. The DNL network where encoded words are embedded with the RCNN-LSTM strategy shows a 61% F1 score, which is 2 points lower than ArchiNet. The configuration of our DNL model has 64 layers, 64 units, epoch size 10, relu activation function, and cross-entropy as loss function [36].

Furthermore, we adopted the best algorithms suggested by Hindle et al. [14] to classify large change commits into five categories, and Soliman et al. [35] to classify architectural discussions. We also explore Naive Bayes (NB), Decision Trees (DT), and Random Forest (RF) [23, 14, 35] for our dataset with the WEKA [13] tool utilizing word-to-vector features [27]. Among them, the most promising classifiers such as NB and DT have less than 55% F1. However, Random Forest (RF), which forms a group of DTs, produces around 58% F1 score for our dataset. The F1 score produced by our technique for the adaptive category is much higher than the competing methods. The ranges of precision and recall rate of ArchiNet among the individual categories are 42.4–77.8% and 64–73.7% respectively, which are more consistent than other classifiers. Notably, from the graph, we can see that F1 scores of RF and DNL for the perfective category is higher than ArchiNet, while significantly lower in the adaptive category because many samples from adaptive might be falsely predicted (high recall rate) into the perfective category (due to lack of handling mechanism of the overlapped concepts). We also see this pattern in the 10-fold validation phase. In this evaluation phase, the distribution of P, R, and F1 scores to all the classes with the test sets indicates a better and stable outcome of ArchiNet with the concept-words.

2) 10-folds Validation: In this phase, we show how our classifier is performing with cross-fold validation since it provides a more accurate evaluation against the over-fitting problem [14, 28]. However, we experiment with the promising methods proven in the first phase. We compare the performance of ArchiNet with DNL and RF by 10-fold cross-validation technique. In 10 iterations, we take 90% samples as the training set, and 10% as the test set exclusively for each of the iterations [28]. The performance comparison is presented in Table V. The F1 score of ArchiNet is around 69%, which is 7 points better than the two classifiers. Deep learning with RCNN-LSTM [19, 36] shows 62% F1 score; RF has a similar outcome as of DNL. F1 scores for some other classifiers are between 50 to 60% with the word-to-vector [27] features. From the median and range values in box plots in Fig. 3 it is observed that the precision and recall rate of ArchiNet (in the ranges 40.1–77.8% and 63.8–73.7% represented by A_p and A_r) are consistent with all the classes (recall is highly consistent than others indicated by DNL_r and RF_r). For the adaptive and corrective categories, the outcome of ArchiNet is significantly higher. Poor recall rate of RF and DNL (marked with circles in Table V) for the adaptive category and high recall rate for the perfective category indicates that many samples from adaptive are falsely retrieved into perfective by both of the classifiers. A similar trend is observed with the corrective category except for lower precision for DNL. Since RF and DNL do not distinguish and select words based on concepts/semantics, they produce more unstable outcomes. In summary, our proposed classifier has better performance for all metrics (F1, P, and R scores) compared to other classifiers because concept-words handle various influential tokens from a commit message efficiently. This exploration answers the research question RQ2.

3) Project-wise Validation: We also conduct cross-projects validation of our proposed approach. We train the classifier with four projects and test with the remaining project in each iteration for the five projects. The project-wise outcomes for both ArchiNet and DNL are presented in Fig. 4. Combined F1 scores of each of the projects produced by ArchiNet are better than that of DNL. None of the project’s F1 scores is below 60% for ArchiNet, while the highest is 69%. The highest precision is 85%, and the recall is 80% (for the perfective and preventive category) for our method. However, the precision and recall can be low for the adaptive category as can be seen in the Fig 4. On the other hand, the adaptive category’s F1 score reaches 62 for the ArchiNet (whereas 23 for DNL). Performance of some of the projects is lower than 10-fold validation because of insufficient training data. Overall, Hadoop’s outcome for...
both ArchiNet and DNL is the most promising because the commit messages in Hadoop could contain a less ambiguous explanation compared to other projects.

4) Sensitivity of Tokens: The performance sensitivity of ArchiNet (for 10-folds) for various token-weights \((w(t))\) combination (in \(1\) and \(2\)) is shown in Fig. 5. The best performance is shown for the combination \(\{t_w, t_p, t_s, t_o\}\) which is three points (69% F1) better than only considering natural terms \(t_p\) (66% F1). Including API, library, and framework name \(t_s\) increments the performance by two points as there is more likely to be an adaptive category for those compared to others. As can be observed from precision and recall in Fig. 5, the adaptive category is the most sensitive. However, we notice that combining source-code operations \(t_o\) affects the performance slightly negatively (66% F1, whereas it is 65% with \(t_w\)); therefore, source code operations are not promising features for classifying the architectural change.

VII. THREATS TO VALIDITY

One of the greatest threats to the validity of our result is that annotating the intention of change is subject to human bias. To reduce this threat, two of the authors independently annotated, and then conflicts are resolved by discussion. Any classifier may suffer an over-fitting problem. To overcome this, we experimented with our classifier with a tenfold cross-validation technique and found a promising result. Another concern of our classification model is how general it predicts change from different programming languages and cross-projects. One of our test sets is collected from Mondal et al. \(29\) that also contains projects of Python language, and have similar outcome as shown in Table VI. A few of the projects such as Hadoop has substantial industrial participation \(17\). Therefore, our study also mitigates generalizability threat to some extent.

Our model can be trained with different metrics. Therefore, for \(2\) in Section \(V\) we also have trained our model with the \(tf-idf\) metrics. However, the result is not as promising as the direct probability value, but still shows a better result than DPLSA, LLDA, and SemiLDA. With this metric, the best F1 scores for the data in \(29\) are 47% and 51% for our benchmark data. Yan et al. \(42\) utilized DPLSA for predicting multiple categories of usual changes (three types). We found only a few of the samples in our data have multiple intentions when architectural changes happened. ArchiNet can handle such scenarios to some extent as we experiment on that mode; when the predicted sample is in Hit@2 \(55, 53\) (within the top 2 ranks), the F1 score is 83.5%. Notably, our proposed classifier is versatile and does not require parameter tuning, unlike others. Our dataset and trained models are available in github.com/akm523/archinet for further investigation.

VIII. RELATED WORK

1) Architectural Design Issues and Solutions Classification: Yamauchi et al. \(41\) proposed a technique considering program identifiers to group the large commits into related components having relations with the functional requirements. An early approach of committed code classification was studied for architectural tactics (design solutions such as resource pooling, secure session management, and so on) \(28\) based on code identifiers (such as heartbeat) mapped with text description (heartbeat emitter and receiver) from a set of trained samples, and commits are predicted using a term-frequency based classifier. Solaiman et al. \(35\) reported Bayesian Network and Naive Bayes as the best algorithms to classify architectural discussions related to six ontology classes (such as technology) into three design steps focusing ambiguous concepts (such as server has different meanings for different cases), concepts expressing reasons of architectural changes are different than those. However, although these classes were either subset or irrelevant to architectural changes, they were not specialized in four architectural changes. In our work, we explore both source code features and concept-token properties to predict the reasons for architecture changes.

2) Architectural Change Classification: We are aware of only one study by Mondal et al. \(29\) to categorize four architectural changes from the text. Their model was generated by popular discriminating feature selection techniques DPLSA \(42\), SemiLDA \(9\), and LLDA \(34\) originally proposed \(42\) for classifying all software changes into three groups, and none of the techniques could handle the twists and challenges of architectural change classification properly. Consequently, the outcome of their proposed technique is poor. Another study by Hindel et al. \(14\) close to ours explored various machine learning techniques for classifying large commits (commits with many files changed) into five groups. We also explore the promising classifiers reported by them: Naive Bayes, Decision Trees, and so on. However, Random Forest (RF), an advanced version of Decision Trees, produces promising

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Perfective</th>
<th>Corrective</th>
<th>Preventive</th>
<th>Adapative</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our data</td>
<td>65</td>
<td>38</td>
<td>68</td>
<td>51</td>
<td>63</td>
</tr>
<tr>
<td>Data (29)</td>
<td>55</td>
<td>61</td>
<td>80</td>
<td>16</td>
<td>63</td>
</tr>
</tbody>
</table>
outcomes with our dataset (but 7 points lower F1 than ArchiNet). Recently, word embedding technique that captures contextual and semantic information with deep learning is being successfully used for software artifacts analysis and classification \([19, 25, 26]\). However, due to the overlapping of concept words, deep learning might not produce the best outcome, which is mostly inexplicable when multiple intentions are required to extract from a single message. Our proposed classifier ArchiNet handles these concerns considering other tokens and gains in performance.

**IX. CONCLUSION**

In this paper, we present a dataset collected from five popular projects and a promising classifier for architectural change categorization from texts. Our study identifies the challenges of classifying changes from both source code properties and textual properties. We address those challenges with a concept analysis approach that indicate the developers’ intentions. Both 10-fold cross-validation and cross-projects validation show that our technique is promising in all aspects compared to traditional methods (F1 score is 70%). We also explore the sensitivity of the performance of our classifier for various tokens. Besides, we extract around 237 keywords (with trained weights for each change category) from the training set. Given the success, many of the text analysis approaches to support the ten activities of software architecture discussed by Bi et al. \([6]\) might be enhanced by adopting our proposed technique. In future, we will explore automatic design documentation generation and architectural versioning schema applying our change classification technique.

**ACKNOWLEDGMENT**

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by two Canada First Research Excellence Fund (CFREF) grants coordinated by the Global Institute for Food Security (GIFS) and the Global Institute for WaterSecurity (GIWS).

**REFERENCES**

Analyzing Open-Source Serverless Platforms: Characteristics and Performance

Junfeng Li\textsuperscript{1}, Sameer G. Kulkarni\textsuperscript{2}, K. K. Ramakrishnan\textsuperscript{3}, Dan Li\textsuperscript{1}
\textsuperscript{1}Tsinghua University, \textsuperscript{2}IIT, Gandhinagar, \textsuperscript{3}University of California, Riverside
Email: hotjunfeng@163.com, sameer.sameergk@gmail.com, kk@cs.ucr.edu, tolidan@tsinghua.edu.cn

Abstract—Serverless computing is increasingly popular because of its lower cost and easier deployment. Several cloud service providers (CSPs) offer serverless computing on their public clouds, but it may bring the vendor lock-in risk. To avoid this limitation, many open-source serverless platforms come out to allow developers to freely deploy and manage functions on self-hosted clouds. However, building effective functions requires much expertise and thorough comprehension of platform frameworks and features that affect performance. It is a challenge for a service developer to differentiate and select the appropriate serverless platform for different demands and scenarios. Thus, we elaborate the frameworks and event processing models of four popular open-source serverless platforms and identify their salient idiosyncrasies. We analyze the root causes of performance differences between different service exporting and auto-scaling modes on those platforms. Further, we provide several insights for future work, such as auto-scaling and metric collection.

Index Terms—cloud computing, serverless, function-as-a-service, characteristic, performance

I. INTRODUCTION

Serverless computing has ushered in a new era in cloud computing. Cloud computing seeks to provide computing and storage services at large scale and low cost to end-users through economies of scale and effective multiplexing \cite{2}. Serverless computing puts multiplexing and scalability to the next level by allowing providers to commit just the required amount of resources to a particular application and utilize the resources for just the time needed to execute an invoked function. Resources are scaled dynamically to meet the demand of user requests. Unlike traditional cloud deployment models, where a number of computing instances are deployed well in advance, serverless computing achieves nearly zero resource cost when there is no demand, and scales to as many instances as needed to meet the traffic demand. Thus, serverless computing could be both scalable and cost effective.

In addition to scalability and multiplexing, serverless computing allows developers to build, deploy and run the application on demand without focusing on server management, according to the Cloud Native Computing Foundation (CNCF) \cite{3}. When an event is triggered, a piece of infrastructure is allocated dynamically for function execution.

In this paper, we significantly add the following work based on a previous workshop version \cite{1} published at WoSC’19: (1) We describe the salient characteristics of four serverless platforms in more detail in §II, including new figure illustration; (2) We elaborate resource/workload-based auto-scaling frameworks in §III; (3) We evaluate four serverless frameworks with more experiments in §IV; we also add detailed analysis about the root cause of performance differences among these serverless frameworks in §IV; (4) We provide insights and future work for serverless platforms in §V.

The underlying details of resource management, \textit{i.e.}, resource allocation, data transmission and function execution, are decoupled from the user. Many cloud service providers (CSPs) offer serverless computing platforms on their public clouds, such as Amazon Web Services (AWS) Lambda, which is an event-driven serverless platform that enables to implement and deploy application in any supported languages and execute on-demand as docker containers. Since public serverless platforms may incur vendor lock-in risk, many open-source serverless platforms spring up and allow developers to freely deploy and manage functions on self-hosted clouds. However, building effective functions requires much expertise and in-depth understanding of platform frameworks and characteristics that affect performance. It is a challenge for a service developer to differentiate and select the proper serverless platform in different scenarios.

To help developers choose suitable open-source platforms to deploy efficient services, we systematically identify and analyze the salient characteristics of several popular open-source serverless platforms (\textit{i.e.}, Knative\textsuperscript{1}, Kubeless\textsuperscript{2}, Nuclio\textsuperscript{3} and OpenFaaS\textsuperscript{4}) and compare their performance. Our key contributions include:

- We provide an understanding of the platform frameworks and interaction between different components of four popular open-source serverless platforms.
- We analyze the salient features of each platform, such as the built-in workload-based auto-scaling mechanism and the event processing model inside the function pod.
- We evaluate the performance of different service exporting and auto-scaling modes, and analyze the root cause of performance gap among different serverless platforms.
- We give several insights for future work, such as auto-scaling and metric collection.

II. BACKGROUND

Many cloud service providers (CSPs) offer serverless computing platforms on their public clouds, such as Amazon Web Services (AWS) Lambda, Google Cloud Functions, Azure Functions and Alibaba Cloud Function Compute. The developers are required to design and deploy their serverless functions based on the supporting services provided by CSPs, such as message queuing, storage and database. Thus it incurs the risk of vendor lock-in. The deployed serverless functions rely

\begin{itemize}
  \item \textsuperscript{1} https://github.com/knative
  \item \textsuperscript{2} https://kubeless.io
  \item \textsuperscript{3} https://nuclio.io
  \item \textsuperscript{4} https://www.openfaas.com
\end{itemize}
TABLE I: Comparison of popular open-source serverless platforms

<table>
<thead>
<tr>
<th>Feature</th>
<th>Nuclio</th>
<th>OpenFaaS</th>
<th>Knative</th>
<th>Kubeless</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queue inside Function Pod</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Support for Multiple Workers in Function Pod</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Function Startup Policy</td>
<td>Warm Start</td>
<td>Cold/Warm Start</td>
<td>Cold/Warm Start</td>
<td>Cold Start</td>
</tr>
<tr>
<td>Service Export Method</td>
<td>Ingress Gateway/NodePort</td>
<td>API Gateway/Ingress Gateway</td>
<td>Ingress Gateway</td>
<td>Ingress Gateway</td>
</tr>
<tr>
<td>Runtime Metric Collection</td>
<td>Metric Server</td>
<td>Metric Server/API Gateway</td>
<td>Metric Server/Queue-proxy</td>
<td>Metric Server</td>
</tr>
<tr>
<td>Auto-scaling Mode</td>
<td>CPU/Memory</td>
<td>CPU/Memory/RPS</td>
<td>CPU/Memory/Concurrency/RPS</td>
<td>CPU/Memory</td>
</tr>
<tr>
<td>Scale-to-zero</td>
<td>x</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
</tbody>
</table>

heavily on specific CSPs, and it is difficult to migrate existing functions to either self-hosted clusters or other public clouds.

The open-source serverless platforms bring more flexibility and allow developers to freely deploy and manage functions on self-hosted clouds. However, there are still some challenges for open-source serverless platforms: (1) it requires a deep understanding of platform features to build effective functions; (2) the developers should manage and maintain serverless platforms by themselves, which requires much expertise of platform frameworks and infrastructures; (3) the performance of open-source serverless platforms may vary in different scenarios, and it is difficult to choose the proper platforms for a specific usage scenario. Therefore, it is necessary to analyze the salient characteristics of popular open-source serverless platforms and compare their performance to help developers choose suitable platforms to deploy efficient services.

III. PLATFORM CHARACTERISTICS

Based on recent popularity, community vibrancy and feature richness, we specifically select four open-source serverless frameworks, i.e., Knative, Kubeless, Nuclio and OpenFaaS, to analyze their characteristics.

A. Dependency on Kubernetes

Kubernetes [4] is a portable and extensible open-source system that facilitates declarative configuration, automating deployment and management for containerized workloads. Most of open-source serverless platforms rely on Kubernetes for orchestration and management of function pods, which are the atomic deployable units in Kubernetes. Fig. 1 shows the pivotal Kubernetes services that serverless platforms depend on. These Kubernetes services are used for: (1) configuration management, (2) service discovery, (3) auto-scaling, (4) pod scheduling, (5) traffic load balancing, (6) network routing and (7) service roll-out and roll-back.

Thanks to the horizontal pod auto-scaler (HPA) feature from Kubernetes, the Kubernetes-based serverless platforms support resource-based auto-scaling. The framework of HPA is shown in Fig. 2. The Kubelet on each node collects the resource metrics of each pod. HPA gets these metrics from the API server. The auto-scaling threshold could be a raw value or a percentage of the pod requested amount for that resource. When the CPU or memory usage of a given function pod exceeds the threshold, HPA automatically triggers the Development controller to scale the pod number.

Fig. 1: Serverless platforms and underlying Kubernetes services.

Fig. 2: Horizontal pod auto-scaling framework.

B. Salient Features of Serverless Platforms

Table I summarizes salient features of four widely-known open-source serverless platforms.

1) Nuclio: The main components of Nuclio are shown in Fig. 3. In each function pod, there is one event listener and multiple worker processes. The event listener receives new events and redirect them to worker processes. Multiple worker processes could work in parallel and improve the performance significantly on a multi-core worker node. The worker process number is set to be static and specified by the configuration file. The open-source version does not have a built-in workload-based auto-scaling feature, but the resource-based auto-scaling is supported by Kubernetes HPA.

Nuclio supports two ways to trigger functions: (1) invoking the function by name through ingress controller, which can distribute the traffic to different back-end pods according to the pre-set load balancing rule (e.g., round-robin, random
and least connection first) and (2) sending requests directly to function pods by NodePort, which is a unique allocated cluster-wide port for the function. In the NodePort method, incoming requests are load balanced at random by Netfilter.

Fig. 3: Nuclio framework.

2) OpenFaaS: The key components of OpenFaaS are shown in Fig. 4. The API gateway provides access to the functions and collects traffic metrics. Faas-netes is the controller for managing OpenFaaS function pods. Prometheus and AlertManager\(^5\) are used for auto-scaling.

Fig. 4: OpenFaaS framework.

Each function pod contains a single container running two type of processes, namely of-watchdog and function process. Of-watchdog is a tiny server that works as the entry-point for incoming requests and forwards them to the function process. Of-watchdog can operate in three modes, i.e., HTTP, streaming and serializing. In HTTP mode, the function process is forked only once at the beginning and kept warm for the entire life cycle of the function pod. In both the streaming and serializing mode, a new function process is forked for every request, resulting in significant cold-start latency and adverse impact on performance. Our evaluation results show that the throughput of the streaming or serializing mode is about \(10^2\) lower than that of HTTP mode.

OpenFaaS has a built-in requests-per-second (RPS) based auto-scaling feature. Prometheus scrapes the traffic metrics from API gateway. AlertManager reads the RPS metric from Prometheus and fires an alert to the API gateway according to the auto-scaling rule defined in the configuration file. Then the API gateway handles the alert and invokes the Faas-netes to scale up or scale down function replicas. Note that the opensource version does not support scale-to-zero feature, which is only available in the commercial version, i.e., OpenFaaS Pro.

3) Knative: Fig. 5 shows the Knative framework. Each function pod consists of two containers, namely queue-proxy container and function container. The queue-proxy is a sidecar container to queue incoming requests and forward them to the function container. The queue-proxy provides a buffer to handle traffic burst in spite of incurring queuing latency. In addition, the queue-proxy collects metrics and expose them via a simple HTTP server, i.e., internal metric server. Multiple workers reside in the user container to process requests in parallel. The communication overhead between queue-proxy container and function container is higher than the process model of Nuclio and OpenFaaS, and thus results in lower performance.

The Knative built-in auto-scaling, i.e., Knative pod autoscaler (KPA), supports both RPS mode and concurrency mode. The auto-scaler scrapes metrics from function pods and computes the replica number based on the auto-scaling algorithm. The deployment controller gets the auto-scaling decision and adjusts the pod number. Knative supports scale-to-zero functionality which recycles all pods of inactive functions. When a new request arrives for an idle function, the ingress controller redirects the request to the activator to buffer it. Then the activator triggers the autoscaler which could scale up the idle function from zero. Once the function is running again, the activator sends the buffered request to the pod. Although scale-to-zero reduces resource usage, it leads to extra cold start latency.

4) Kubeless: Kubeless is another open-source platform built on top of Kubernetes. Fig. 6 describes the key components and the working model of Kubeless. There are several options for ingress controllers. We experiment with Nginx ingress controller\(^7\) and Traefik ingress controller\(^8\), and opt for Traefik due to better performance. Kubeless leverages Kubernetes HPA for auto-scaling and does not support scale-to-zero.

Fig. 5: Knative framework.

C. Service Exporting and Network Routing

1) Service Exporting: The function pods are dynamic entities that can be created and destroyed at any time due to auto-scaling, failures, etc. Hence, Kubernetes provides service as an abstraction to access the pods of the same function. There are several ways to export services: (1) the service could be assigned a NodePort, which is used to route the

---

\(^5\) https://prometheus.io/\(^6\) https://github.com/prometheus/alertmanager

\(^7\) https://kubernetes.github.io/ingress-nginx\(^8\) https://traefik.io
incoming traffic to the entry node and let kernel stack control load-balancing of the traffic across active pods; (2) the API gateway/ingress controller works as the entry point and the services are exported with specific URLs. The API gateway/ingress controller component of the serverless platform can be accessed from outside the cluster by a external public IP address. Once the API gateway/ingress controller receives an incoming request, it determines the service for the request according to the URL, and then load-balances and routes the packet to a back-end active pod instance.

Fig. 7: Network routing (Flannel mode) to export the services.

2) Network Routing: Fig. 7 describes Flannel - a simple Kubernetes overlay networking framework to route the traffic to function pods. The Kube-Proxy pod is responsible for setting up the routing and load-balancing rules, i.e., the netfilter destination network address translation (DNAT) rules to change the destination IP of incoming request packets [5]. The Kube-Flannel pod is responsible for intercepting the packets and performing UDP encapsulation/decapsulation for the traffic exiting/entering the physical network interface.

IV. PERFORMANCE EVALUATION

We first compare the overall performance of different open-source serverless platforms. Based on the performance results of multiple service exporting modes and auto-scaling modes, we analyze the root cause of performance gap among different serverless platforms.

A. Experimental Setup and Workload Description

We evaluate the serverless platforms on the CloudLab testbed [6] consisting of one master and two worker nodes, each of them equipped with Intel CPU E5-2640v4@2.4GHz (10 physical cores), running Ubuntu 16.04.1 LTS (kernel 4.4.0-154-generic). We build all four serverless platforms on Kubernetes (v1.20.0), using the latest version available at the end of the function runtime; (4) when the response is sent out of the pod) vary. Several serverless functions of Python 3.6 runtime are implemented. We use wrk [10] to generate HTTP workloads for invoking serverless functions.

9 Nuclio (v1.6.1); OpenFaaS (v0.20.11) with HTTP mode of-watchdog; Knative (v0.21) with Istio ingress controller (v1.8.4); Kubeless (v1.8.8) with Traefik ingress controller (v2.4).

10 https://github.com/wg/wrk

B. Performance

1) HTTP Workload: To evaluate the baseline performance of different serverless platforms, we implement a HTTP workload function that could fetch a four-byte static webpage from a local HTTP server on the master node. For a fair comparison, we limit to a single instance of the function pod, disable auto-scaling and configure the same queue size and timeout parameters (50K requests, and 10s timeout) at the ingress gateway and function pod components across all the platforms. For Nuclio and Knative, we further restrict it to a single worker in one pod. Every experiment lasts for two minutes and we measure for one minute after one-minute warm-up. The experiment is repeated for 20 times. Fig. 8 shows the throughput for varying number of concurrent connections and the latency profile for concurrency level of 100. Nuclio has the least 99%ile latency within 500ms, as it allows queuing only within the function pod, while OpenFaaS and Knative can queue requests at ingress/gateway components. OpenFaaS shows heavy tail due to queuing at both the gateway and of-watchdog components. Kubeless drops the connections at the ingress, resulting in additional retries from the client and hence lower throughput. The latency with Kubeless is lower because there is no queue inside the Kubeless function pod.

Fig. 8: Performance of HTTP workload function. Error bars indicate standard deviation over 20 runs.

2) Latency Breakdown of Single Request: We analyze the delay overheads incurred in processing serverless functions for different platforms. We breakdown the processing delays within the function pod. For this experiment, we use curl to send one request for hello-world function and use tcpdump to capture the packets on the worker node of the function pod. We record four timestamps, i.e., (1) when the request reaches the function pod; (2) start of the function runtime; (3) end of the function runtime; (4) when the response is sent out of function pod. The experiment is repeated for 20 times and the average time intervals between these timestamps are shown in Fig. 9. In all frameworks, the actual run-time of the function (0.001ms) is the same. However, the function initiation time (time taken for request to be forwarded to the function instance) and function response delay (time taken for the response of the function to be sent out of the pod) vary. This depends on how the data is packaged and shared with the function instance. Due to forking-per-request, Kubeless
incurs very high delay in forwarding the packet to the function instance.

<table>
<thead>
<tr>
<th>Process</th>
<th>1→2</th>
<th>2→3</th>
<th>3→4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nuclio</td>
<td>0.63</td>
<td>0.001</td>
<td>0.54</td>
</tr>
<tr>
<td>OpenFaaS</td>
<td>1.32</td>
<td>0.001</td>
<td>0.93</td>
</tr>
<tr>
<td>Knative</td>
<td>1.30</td>
<td>0.001</td>
<td>0.62</td>
</tr>
<tr>
<td>Kubeless</td>
<td>4.96</td>
<td>0.001</td>
<td>2.63</td>
</tr>
</tbody>
</table>

Fig. 9: Latency breakdown of function execution (ms).

C. Auto-scaling

To study the auto-scaling capabilities provided by different serverless platforms, we compare the features of both the workload-based and resource-based auto-scaling under different workload characteristics. We use the same HTTP workload function as in §IV-B1.

1) Workload-based Auto-scaling: Both Knative and OpenFaaS support workload-based auto-scaling. While the workload-based auto-scaling metric of OpenFaaS is RPS, the metric in Knative is concurrency, i.e., the concurrent request number. For a fair comparison, we set equivalent auto-scaling configuration parameters for these platforms. We use work to send a steady flow of requests (with concurrency of 100 and RPS of 100) and run the experiment for 60s. Periodically every 2s, we monitor the number of pod instances, CPU and memory usage, and throughput. From Fig. 10, we observe that Knative scales multiple instances at a time to reach 10 instances quickly (in 12s), while OpenFaaS just scales up one instance at a time, taking 26s to scale up to 10 instances. Due to the longer process chain of auto-scaling in OpenFaaS (i.e., API gateway → Prometheus → AlertManager → API gateway → Faas-netes), the scaling latency of OpenFaaS is higher than that of Knative. Although the CPU usage for the scaled instances looks identical, the memory pressure of Knative is higher. This stems from the differences in python runtimes and proxies (i.e., the queue-proxy in Knative and of-watchdog in OpenFaaS).

2) Resource-based Auto-scaling: All these Kubernetes-based platforms support resource-based auto-scaling. In this experiment, we use CPU usage as the metric of HPA and the CPU threshold is set to be 50%. The other experiment configurations are the same as those in §IV-C1. As Fig. 11 shows, except for Kubeless, the auto-scaling behavior is similar across all the platforms, i.e., auto-scaling tries to double the instances at each step until it reaches the maximum. However, the duration of each step depends on the CPU utilization factor, which in turn depends on the serverless platform specific components, such as event-listener, of-watchdog and queue-proxy. Nuclio, being relatively more CPU hungry, is able to scale more rapidly (in 40s) than Knative and OpenFaaS. For Kubeless, the fork-per-request and no queuing of function pods result in high latency and packet loss, which in turn contributes to lower throughput and lower CPU utilization [7]. Thus it leads to poor auto-scaling performance.

V. INSIGHTS AND FUTURE WORK

1) Promising Platform: In spite of the moderate performance compared with other open-source platforms, Knative has many useful features, such as scale-to-zero and multiple auto-scaling modes, and active community that can provide lots of help for users and developers. Thus Knative is a suitable platform for further development and innovation in serverless computing.

2) Auto-scaling: For current auto-scaling mechanisms, such as workload/resource-based auto-scaling, the auto-scaling metric and threshold are set by tenants. Because the tenants may not really know the runtime characteristics of their functions (i.e., resource usage and execution time), they easily mis-configure the auto-scaling settings. In addition, it is not always easy to predict the correct indicators that could show whether the current function pods are under-resourced or under-utilized. Thus a more smart auto-scaling algorithm is needed to be designed to both properly meet the workload demand and save the resources in different scenarios.

---

12 In Knative, we set the minScale and maxScale instances as 1 and 10, target to 10, max-scale-up-rate to 100, tick interval to 2s, and stable window to 10s. Likewise, for OpenFaaS, we set scale-factor to 10 and configure the alert-notification window to 2s, and RPS threshold to 10.
3) Function Startup Policy: There are two options for function startup policy, i.e., cold start and warm start. For the functions with low invocation rates, cold start policy could help reduce resource usage in the case of no incoming requests, but it leads to extra cold start latency. Therefore, the cold start is not appropriate for time-sensitive functions [8]. We should carefully choose the function startup policy in accordance with the scenarios and user demands.

4) Metric Collection: The on-demand provisioning feature of serverless computing depends on several mechanisms, such as auto-scaling, scheduling and load balancing. All these mechanisms leverage metrics to make the decision. Many platforms, such as Knative and OpenFaaS, use scraping method to fetch metrics from pods. In our experiments, we find that the scraping method may leads to large traffic overhead when there are a large number of pods. Using sampling to only scrape a section of pods can partly decrease the overhead. However, sampling may miss out the abnormal pods and reduce the accuracy of metrics. Hence, a more efficient metric collection mechanism is worth studying further.

5) Service Export and Network Routing: There are many ways to export services and route incoming requests to backend function pods, such as cluster IP, NodePort, function name/URL. All of them have both strengths and weaknesses, and should be chosen with caution.

6) Function Chain: It is useful to chain multiple functions for stateful workflows and complex services. How to make function chain more efficient and powerful needs to be explored in future work.

VI. RELATED WORK

In work [9]–[11], the authors conducted several measurements on different cloud serverless platforms (AWS Lambda, Microsoft Azure, Google Cloud), and found the AWS to be better in terms of throughput, scalability, cold-start latency. The work [12], [13] investigates the different factors that influence the performance of AWS Lambda, namely the impact of the choice of language of the function, memory footprint of the function, etc. Work [14] evaluates the performance of Fission, Kubeless and OpenFaaS serverless frameworks and characterizes the response time and the ratio of successfully completed requests for different loads. However the work fails to characterize the throughput of these platforms and accounts for the mean latency (response time) and successful responses at different load characteristics, which is debatable, without the proper consideration and configuration of the serverless platform specific configuration parameters, resulting in inaccurate results. Work [15] quantitatively evaluates Apache OpenWhisk, OpenFaaS, Kubeless, and Knative platforms. The results for Kubeless are similar, but for the other platforms, we feel the presented results are inaccurate. This could be due to the usage of Kubernetes. In contrast, our work focuses on discerning the architectural blocks that impact the performance of Kubernetes-based open-source serverless platforms.

VII. CONCLUSION

We elaborate the working models of different popular open-source serverless platforms and identify their key characteristics. In addition, we analyze the root causes of performance gap of different service exporting and auto-scaling modes on those platforms. Further, several insights are proposed for future work, such as auto-scaling, service export and metric collection.

Acknowledgment: This work was supported by National Key Research and Development Program of China under Grant 2018YFB1800500, and US NSF grants CRI-1823270, CNS-1763929, and grants from Hewlett Packard Enterprise Co., Futurewei Technologies Inc., and China Scholarship Council. Dan Li is the corresponding author.

REFERENCES

I-CMOMMT: A multiagent approach for patrolling and observation of mobile targets with a continuous environment representation

Jamy Chahal$^{1,3}$ Assia Belbachir$^{1,3}$ Amal El Fallah Seghrouchni$^{1,2}$

$^1$ Sorbonne Université, LIP6 - UMR 7606 CNRS
$^2$ Université Mohammed VI Polytechnique
$^3$ Ai Movement - International Artificial Intelligence Center of Morocco
Institut Polytechnique des Sciences Avancées, Department of Aerospace System, Ivry-Sur-Seine, France

Abstract

Agent-based modelling has been widely studied for observing moving targets and patrolling. However, in general, the studies are interested either in observation in a continuous environment or patrolling in a graph representation. In order to deal jointly with observation and patrolling, a common representation of the environment is required. In this paper, we firstly proposed a new environment representation’s formalism, merging both agent-based distributed patrol and observation method. Secondly, we implemented a new approach called I-CMOMMT to cope with a trade-off between observation and patrolling using our new formalism. The obtained results are compared with other methods to show the efficiency of our approach.

1. Introduction

Multiagent paradigm is widely used for mobile targets observation and patrolling. The observation problem consists in positioning agents to maximize the number of viewed targets by at least one agent. The Cooperative Multi-Robot Observation of Multiple Moving Targets (CMOMMT) is a well defined problem [1]. It is composed by a team of $m$ robots, supervising $n$ targets into an enclosed spatial region, where the number of robots is greater than the number of targets. To maximize the number of viewed targets, the author [1] proposed a distributed solution called A-CMOMMT. This solution is based on local force vector, where each agent is attracted by all the targets within its sensor range, and repulsed by all other agents within a protection range. The authors in [2] proposed the Behavior CMOMMT (B-CMOMMT) to improves the target’s distribution among the agents, by adding for instance the ability for the agents to ask for help if they are facing a potential loss of target tracking.

The patrolling problem consists of positioning agents to minimize the idleness, which represents the time difference between two visits of a same location by at least one agent. Idleness was formalized using a graph representation by the authors in [3]. An area is represented by a node and an idleness to be minimized. The authors [4] put forward the use of Long short-term memory (LSTM) to learn a patrolling strategy. In [5], the authors consider an open system, where agents can enter/leave the system. The coordination is based on auctions, where agents can trade their belonging nodes.

On one hand, the observation problem uses a continuous spatial representation to incorporate the motion of targets and agents. On the other hand, the patrolling problem is based on graph representation to evaluate the idleness of a node. To deal with both observation and patrolling problem, we developed a conceptual framework unifying the two representations into a continuous environment representation. Besides, we developed a method called Idleness CMOMMT (I-CMOMMT), to consider both patrolling and observation, and we assess it through new patrolling evaluation tools.

The paper is organized as follows. Section 2 explains our conceptual framework. Section 3 defines our developed method I-CMOMMT. Section 4 describes our experiments and the obtained results. Section 5 concludes our work.

2. Conceptual framework

We define the patrolling and target tracking problem as follows: $S$ is a two dimensional enclosed area, where $S \subseteq \mathbb{R}^2$. $A$ is a set of $m$ patrolling agents. $O$ is a set of $n$ targets.

Each patrolling agent $a_i \in A$ (with $i \in [1, m]$) is defined by a set of three parameters $(state, obs, com)$: state contains the Cartesian position $p_{ai}$ of the agent $a_i$, and the speed $v_{ai}$, where $v_{ai} < v_{dmax}$, obs is described by an observed surface $s_{oi} \in S$ and the sensor’s description. The sensor’s description includes the percentage of false positive and false negative detection and the time processing. com defines the agent’s communication capabilities, described by a surface $s_{ci}$ and the communication’s limitation. The latter includes the delay and the bandwidth’s constraint.

Each target $o_j \in O$ (with $j \in [1, n]$) is defined by a
state. For the whole mission of patrolling and observation, the objective is to minimize the average idleness (eq. (4)), as well as the maximum idleness, defined in eq. (5), and to maximize the target’s observation through the $A$ metric (eq. (7)). The following subsections will describe in detail these metrics and our work on changing the environment representation into a continuous one.

2.1. From a graph to a continuous environment representation In a graph representation, we define $i_k(t)$ the idleness of a node $n_k$ at a time $t$. At each time step $\Delta t$: $i_k(t + 1) = i_k(t) + \Delta t$. If, at a time $t$, the node $n_k$ is observed by at least one agent $a_i$, then $i_k(t) = 0$. However, the graph representation is not suitable for agents and targets evolving in a continuous environment. We propose to create a continuous idleness function called $I(x, y)$. This function returns the idleness of a position in two dimensions $(x, y)$ defined by: $\mathbb{R}^2 \to \mathbb{R}^+$, $x, y \mapsto I(x, y)$ . At the beginning of the mission, the idleness of the whole map is equal to 0:

$$I_{t=0}(x, y) = 0$$

(1)

Then, at each time step $\Delta t$, the idleness of each position changes such as:

$$I_{t+1}(x, y) = I_t(x, y) + \Delta t$$

(2)

When a patrolling agent $a_i$ observes an area $s_o$, then the region’s idleness changes to 0:

$$\forall(x, y) \in s_o : I_t(x, y) = 0$$

(3)

2.2. Evaluation criteria In order to compare different developed approaches for patrolling and observation, several evaluation criteria need to be defined. In the following sections, we define them for the patrolling and then for the observation context.

2.2.1. Patrolling problem In [3, 6], the authors propose several criteria resulting from the notion of idleness. They are represented on the left side of the table 1. In order to lighten the memory of the agent, we propose to discretize the continuous idleness function into multiple cells. Let $d_f$ be the discretization factor and $M_e$ the chosen metric (e.g. meter, km). Then each surface $M_e^2$ is transformed into $d_f^2$ cells.

The discretization factor $d_f$ has to take into account the surface’s observation $s_o$ of the agents. We propose to choose a $d_f$ such as $s_o$ covers at least $3 \times 3$ cells. The size of $d_f$ must also be dimensioned considering the processing capacity of the agent. This discretized representation is on the right side of the table 1.

The objective of a patrolling method is to minimize the idleness’s average, which can be done in the continuous representation case with the defined equation (4). However, this criterion is an average, and does not reflect whether a region has been neglected for a long time.

In [7], the author underlined other different patrolling evaluation criteria, in particular to consider the maximum idleness of a node during the whole duration $T$: $i^{\text{max}}_{G} = \max_{t \in [0 : T]} I^{\text{av}}_{G}(t)$ We propose to use the same definition in the context of a discrete map as follows:

$$i^{\text{max}}_{M} = \max_{t \in [0 : T]} I^{\text{max}}_{M}(t)$$

(5)

However, this criterion is still not significant. Indeed, by using a discretized map idleness, a node (which represents an area) cannot be compared to a cell (which represents the smallest surface unit). Therefore, the maximum cell idleness (eq. 5) can easily reach the mission duration $T$. Thus, instead of evaluating the maximum idleness of a cell, we propose in this paper to consider a new criteria: the maximum idleness of a set of cells, called region. In order to fit the aggregation of cells with the agent’s observation capabilities, we set the region’s surface equals to the observation surface $s_o$. As mentioned in [8], for the image processing field, getting the average intensity of pixels in an area is performed by filtering. This transformation is done by using a two-dimensional convolution, based on a kernel $\omega$. By analogy, the map $M$ is a matrix, composed by intensities (in this case, idleness), that we can filter using the following equation: $M_r = \omega * M$. With $M_r$, the regional map, containing the region’s idleness $i_{rk}(t)$ for the cell $c_k$. Then, we can compute the maximum region idleness $i^{\text{max}}_{M_r}$ by:

$$i^{\text{max}}_{M_r} = \max_{t \in [0 : T]} \max_{c_k \in C} i_{rk}(t)$$

(6)

In order to perform an average, the kernel $\omega$ is a square ma-

<table>
<thead>
<tr>
<th>Graph</th>
<th>Discrete map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst graph’s idleness</td>
<td>Worst map’s idleness</td>
</tr>
<tr>
<td>$i^\text{av}<em>G(t) = \max</em>{n_k \in N} i_k(t)$</td>
<td>$i^\text{av}<em>M(t) = \max</em>{c_k \in C} i_k(t)$</td>
</tr>
<tr>
<td>with $i_k(t)$ the idleness of the node $n_k$ and $N$ the set of nodes in the environment.</td>
<td>with $i_k(t)$ the idleness of the cell $c_k$ and $C$ the set of cells in the environment.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instantaneous graph idleness</th>
<th>Instantaneous map idleness</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i_G(t) = \frac{1}{</td>
<td>N</td>
</tr>
<tr>
<td>with $</td>
<td>N</td>
</tr>
</tbody>
</table>

Table 1: Idleness’s definition using a graph (left) and a discrete map (right) representation.

In [8], for the image processing field, getting the average intensity of pixels in an area is performed by filtering. This transformation is done by using a two-dimensional convolution, based on a kernel $\omega$. By analogy, the map $M$ is a matrix, composed by intensities (in this case, idleness), that we can filter using the following equation: $M_r = \omega * M$. With $M_r$, the regional map, containing the region’s idleness $i_{rk}(t)$ for the cell $c_k$. Then, we can compute the maximum region idleness $i^{\text{max}}_{M_r}$ by:

$$i^{\text{max}}_{M_r} = \max_{t \in [0 : T]} \max_{c_k \in C} i_{rk}(t)$$

(6)
trix, with the size of the observation surface in the discrete representation, made only by ones over the sum of element.

Therefore, a good patrolling strategy tends to minimize the idleness of regions through the minimization of the equation (6).

2.2.2 Observation problem The observation evaluation criteria has already been defined in the formalization of the CMOMMT [1]. We propose to use the same notation and definition. The metric $A$ represent the number of targets seen on average by at least one agent during evaluation time $T$. Therefore, the objective of an observation strategy is to maximize the metric $A$.

3. I-CMOMMT Proposal

We propose to combine observation and patrolling by developing a distributed method called Idle-ness CMOMMT (I-CMOMMT), based on the force field of the A-CMOMMT. In this method, an agent $a_i$, at a time $t$, undergoes a force from each target $j$ under its observation surface (weighted by $\omega_j$) and from each other agent $k$ within its communication range:

$$F(a_i, t) = \sum_{j=1}^{n} \omega_{ij} f_{ij}^p + \sum_{k=1}^{m} f_{ik}^p$$ (7)

The magnitudes of $f_{ij}^p$ and $f_{ik}^p$ are defined in the figure 2 and 3 of the paper [1], alongside with the parameter $d_{o1}, d_{o2}, d_{o3}, d_{r1}, d_{r2}$ and the concept of predictive tracking range. The weights $\omega_{ik}$ depends on the strategy design.

Based on the equation (7), we propose to add a force $f_i^p$ related to the patrol problem. This force is weighted by the value $\lambda(t) \in [0; 1]$, implying that $\lambda(t)$ is the patrolling level priority over the observation at a time $t$. Then, the sum of the force is defined by:

$$F(a_i, t) = (1 - \lambda(t)) \sum_{j=1}^{n} \omega_{ij} f_{ij}^p + \sum_{k=1}^{m} f_{ik}^p + \lambda(t) f_i^p$$ (8)

If the agent $a_i$ has no target under its observation surface $s_o$ at a time $t$, then $a_i$ is doing only patrolling with $\lambda(t) = 0$. Besides, we consider the weights $\omega_{ij} = 1$.

Because each scenario is unique, we let the experimenter define the idleness’s indicator $\sigma$ (in seconds) from which the idleness of a region is considered to be high. The desired agent’s behavior of the I-CMOMMT is to perform observation as long as the idleness of regions is low. When the idleness of at least one region approaches the $\sigma$ value, $\lambda(t)$ increases to prioritize the patrolling.

Therefore, $\lambda(t)$ has to evolve according to the maximum idleness of region at a time $t$: $\max_{c \in C} ir_k(t)$. We propose the following definition, by using the $\tanh$ function to keep $\lambda(t) \in [0; 1]$: $\lambda(t) = \tanh(\frac{\max_{c \in C} ir_k(t)}{\sigma})$

With $ir_k(t)$ the region’s idleness at the cell $k$ and $C$ the set of cells in the environment. The force $f_i^p$ is directed toward the highest region’s idleness only, with a constant magnitude of 1. This direction is changed when another region has a greater idleness. In case of multiple regions having the same idleness, the agent randomly selects only one among them.

To improve the patrolling strategy, we propose that each agent shares its own map with all the others agents belonging to its communication surface. During the reception, the agent updates its map by choosing the most up to date information, which is the minimum between its own cells and the ones received.

4. Experimentation

Several simulations have been carried out to evaluate the effectiveness of our proposed I-CMOMMT method in the context of patrolling and observation mission. For this purpose, we defined a random target’s behavior. The target is randomly choosing a position from the environment $S$ with a constant velocity, and then randomly selects a new position. We also suppose that there is no communication between the targets, nor collision consideration.

The experiments have the following configuration: An environment’s surface of $75m \times 75m$, which is disretized by a factor $d_f = 3$ cells/m. The experiment duration is $T = 1$ 800s. Agents have an observation’s range of 4m and a communication’s range of 5m. Besides, the target’s maximum speed $v_{o_{max}} = 0.5m/s$ and the agent’s maximum speed $v_{a_{max}} = 1m/s$. Finally, we set $\sigma = 0.8 \times T$, $d_{o1} = 1m$, $d_{o2} = 2m$, $d_{o3} = 4m$, $d_{r1} = 1m$, $d_{r2} = 2m$ and a predictive tracking range of 5m.

In our experiments, we consider that communication and observation are only limited by the range (implying no delay, nor bandwidths constraint and perfect target’s detection). We compare the I-CMOMMT method with the observation’s strategy A-CMOMMT and three other patrolling strategies. Inspired by the work of [3], we adapt these strategies in our continuous idleness function case:

**Random Reactive (RR)**: The agent randomly selects a position, goes there, and randomly selects a new one. In [3], the agent randomly selected a node.

**Closest Idleness (CI)**: The agent chooses, among the surrounding cells, the one with the highest idleness. The agent can perform a map sharing with other agents under its communication range. In [3], the agent selected the surrounding nodes, without communication. The behavior was called the Conscientious Reactive.

**Highest Idleness (HI)**: The agent chooses, among all the cells, the one with the highest idleness. The agent can per-
form a map sharing with other agent under its communication range. In [3], the agent selected among all the nodes, without communication. The behavior was called the Conscientious Coordinated.

We have run 15 experiments for each set of agents and targets configuration. The statistical obtained results of the evaluation criteria are shown in Figure 1 and Figure 2.

![Figure 1](image1.png)

**Figure 1:** Illustration of the obtained results from the Map idleness and the ratio number of agents and targets.

The map idleness for the aforementioned five methods are shown in Figure 1. Through this figure, we can see that the highest average idleness is obtained from the A-CMOMMT. In contrast, we obtained a better minimization from the patrol-oriented methods (such as CI and HI). The I-CMOMMT method is an in-between, it improves the map idleness compared to the A-CMOMMT, but it is not as efficient as the patrol-oriented methods.

![Figure 2](image2.png)

**Figure 2:** Illustration of the obtained results comparing A metric according to agent and target ratio.

Figure 2 compares the A metric for the aforementioned five methods. On one hand, the patrol-oriented methods have no interest in the observation objective, leading to a low value of A. On the other hand, A-CMOMMT is focusing on the observation, with a high value of the metric A. Therefore, the I-CMOMMT is an in-between method, by considering observation as well as patrolling. In our scenarios, on average, the I-CMOMMT reaches 71% of the A-CMOMMT observation’s efficiency. While it reduces, still in comparison with the A-CMOMMT, 25% of the average map idleness.

From these experiments we can consider I-CMOMMT as a method that makes a compromise between both observation and patrolling problem.

## 5. Conclusion and future work

In this paper we are interested on combining approaches related to observation and patrolling. The patrolling representation and analysis tools are based on graph whereas the observation problem uses a continuous representation. In this work, we proposed to merge both representations into a uniform continuous representation. This transformation has been achieved through the use of a continuous idleness function \(I_t(x, y)\).

A new patrolling and observation method called Idleness CMOMMT (I-CMOMMT) was proposed. This method weights patrolling and observation forces to find a good balance between the two problems. We evaluate I-CMOMMT with different methods. Our result shows that the I-CMOMMT approach is successfully achieving its goal to consider both patrolling and observation, by being more efficient for patrolling than observation-oriented method and observing more targets than patrolling-oriented methods.

## References


NVMSorting: Efficient Sorting on Non-Volatile Memory

Zhaole Chu, Yongping Luo, Peiquan Jin, Shouhong Wan
1School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
2Key Lab. of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei, China
jpq@ustc.edu.cn

Abstract—Non-volatile memory (NVM) as a new type of storage technology has many advantages such as non-volatility, byte addressability, high storage-density, and low energy consumption. Meanwhile, NVM has some limitations, e.g., asymmetric read and write latency, limited write endurance, and high price. Therefore, at present, it is not realistic to completely replace DRAM with NVM in computer systems. A more feasible scheme is to adopt the hybrid memory architecture composed of NVM and DRAM. Following the assumption of hybrid memory architecture, this paper proposes an NVM-friendly sorting algorithm called NVMSorting. Particularly, we introduce a new concept called natural runs to improve the existing MONTRES algorithm and present the cost analysis of the algorithm in the hybrid memory architecture. In order to verify the performance of our proposal, we implement six existing sorting algorithms as baselines, including the MONTRES algorithm, and conduct comparative experiments on an unsorted dataset and a partially sorted dataset. The experimental results suggest the efficiency of NVMSorting in terms of execution time and NVM writes. Especially, on the partially sorted dataset, NVMSorting has 6.2% improvement on time performance and 5.7% reduction on NVM writes compared to MONTRES, and 13.0% performance improvement and 27.1% NVM-write reduction compared to the traditional merge sorting algorithm.

Keywords—Non-Volatile Memory, Hybrid Memory, Sorting Algorithm

I. INTRODUCTION

Non-volatile memory (NVM) such as Phase Change Memory (PCM) is one of the research hotspots in recent years, and is also considered as a hot candidate for the next generation of storage technology. NVM has some special properties [1]–[3]. First of all, differing from DRAM, it is non-volatile, meaning that all data written into NVM will not be lost when the host computer is shut down. Second, differing from magnetic disks or solid-state drives (SSD) that only support block-based data accesses, NVM is byte addressable, which is similar to DRAM. Third, the density of NVM is generally higher than that of DRAM and is comparable to that of SSD. To this end, NVM has the advantages of both disks and DRAM. However, NVM also has some limitations compared to DRAM and disks. Firstly, the read and write latency of NVM is not balanced. Particularly, NVM has the similar read latency as DRAM, but its write latency is higher than that of DRAM.

In addition, the endurance of NVM is limited, meaning that after a certain number of writes (10^8 at present), NVM will become unstable. Thus, algorithms running on NVM have to be write-friendly. In summary, we list the main features of SSD, DRAM, and NVM in Table I.

Due to the limitations of NVM, currently it is not realistic to completely replace DRAM with NVM. A more feasible scheme is to adopt the hybrid memory architecture composed of NVM and DRAM. There are two kinds of hybrid memory architectures. The first type is the hierarchical architecture [2], which takes DRAM as the cache of NVM. In this architecture, only the DRAM space is recognized by the operating system as main memory and does not utilize the high density of NVM. In addition, it will introduce additional cost of data migration and consistency. The second type is parallel architecture [3], in which NVM and DRAM are both used as the main memory. In this way, we can make good use of the advantages of NVM, such as byte addressability and persistence. In addition, we can reduce the write operations to NVM by devising appropriate algorithms. Therefore, so far, the parallel architecture has received much attention in NVM-related research. In this study, we also adopt the parallel architecture.

Because of the emergence of NVM with high storage density, we can use NVM to replace the traditional disk as the persistent storage device and build a hybrid memory system without disk. Based on this inference, this paper studies the sorting algorithm in hybrid memory. At the same time, due to the read/write asymmetry of NVM, we need to reduce the writing operations of sorting as much as possible. Briefly, we make the following contributions in this study:

(1) We improve the existing MONTRES algorithm to make it suitable for NVM. MONTRES was designed as an external sorting algorithm on SSD. In this paper, we propose a new

<table>
<thead>
<tr>
<th>Table I</th>
<th>Comparison of SSD, DRAM, and NVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SSD</td>
</tr>
<tr>
<td>Read Latency</td>
<td>25µs</td>
</tr>
<tr>
<td>Write Latency</td>
<td>500µs</td>
</tr>
<tr>
<td>Endurance</td>
<td>10^5</td>
</tr>
<tr>
<td>Density</td>
<td>High</td>
</tr>
<tr>
<td>Non-Volatile</td>
<td>Yes</td>
</tr>
<tr>
<td>Byte-Addressable</td>
<td>No</td>
</tr>
</tbody>
</table>
concept of natural runs and devise a new NVM-friendly memory sorting algorithm called NVMSorting. Compared to the original MONTRES algorithm, the proposed NVMSorting can detect partially ordered runs and reduce the sorting cost.

2) We present cost analysis of the NVMSorting algorithm to theoretically demonstrate that NVMSorting has a lower cost than existing sorting algorithms.

3) We experimentally compare the time performance and the number of NVM writes of the NVMSorting algorithm with six existing sorting algorithms. The experimental results show that NVMSorting has better time performance and less NVM writes than existing sorting algorithms. In addition, NVMSorting achieves better performance on the partially ordered dataset, indicating that it can effectively detect the partially ordered runs.

II. RELATED WORK

To the best of our knowledge, few studies have been focused on the improvement of fundamental sorting algorithms on NVM.

The first study [4] presented a write-limited sorting algorithm within the context of database query processing. It proposed three write limited sorting algorithms, namely segment sort, hybrid sort and lazy sort. These algorithms are all based on trading writes for reads to achieve an optimal NVM memory cost. They either offer a knob to adjust the read and write ratio of an algorithm, or use a lazy mechanism to delay result materializing until the penalty outweighs the gains.

The second work [5] proposed a cost model for sorting on storage devices with asymmetric read and write latency. In this literature, the authors presented three sorting algorithms (merge sort, sample sort, and heap sort using buffer trees) for the asymmetric external memory model and gave a detailed cost analysis. However, this work is towards page-based storage devices, such as flash-memory-based SSDs. Although flash memory also has limited write endurance and low write latency, it is much different from NVM, because NVM can be used as main memory while flash memory can only be used as secondary storage.

In the literature [6], the authors proposed a write-once sorting algorithm, named B*-sort. Differing from previous work, this work focused on pure NVM-based embedded system. Unlike the commonly used array-based sorting algorithms, the B*-sort adopts a new concept, tree-based sort which is inspired by the binary search tree that has the write-once property during tree construction. The algorithm can guarantee \(O(n)\) writes. Because of the tunnel-list structure proposed by the author, B*-sort can also guarantee \(O(n\sqrt{n})\) reads. Although the pure NVM memory system could be realistic in the future, many studies have reported that the hybrid memory architecture will be more realistic in next years. This is mainly because that the DRAM's speed is higher than that of NVM. Our study is also towards the hybrid memory architecture. Therefore, we will not compare the B*-sort algorithm in the experiment.

Recently, Luo et al. [7] proposed an optimal data placement model for solving the data placement issue on DRAM-NVM-based hybrid memory. They also developed a new sorting algorithm that adopted heap structures to accelerate the sorting process. However, this sorting algorithm was an in-memory sorting one that ran on the parallel memory architecture composed of DRAM and NVM, which is different from the memory architecture of this study.

III. DESIGN OF NVMSORTING

A. Motivation

NVM and flash memory have similar defects, which is the read/write asymmetry and the limitation of endurance. Based on this observation, we can infer that a write efficient algorithm designed for SSD may have the same effect when it is transplanted to NVM. At present, the research of sorting algorithms for SSD has made great progress, and a state-of-art sorting algorithm for SSD is called MONTRES [8]. Therefore, this study aims to improve MONTRES to make it suitable for NVM-based hybrid memory architecture.

MONTRES was designed as an external sorting algorithm on SSD. Its main idea is to persist data to SSD as early as possible so that we need not partially write data to intermediate runs. To achieve this, they proposed the merge-on-the-fly mechanism and the run-expansion mechanism to generate runs as large as possible. MONTRES has been demonstrated to be helpful for accelerating the phase of merging runs. However, MONTRES was designed for paginated SSDs and is not suitable for byte-addressable NVM.

In the literature [9], researchers proposed the MONTRES-NVM algorithm to optimize MONTRES for NVM, which offers to detect all the sorted pieces in the original data set. Each sorted piece is treated as a sorted run and it is ignored in the run generation phase, to reduce useless NVM writes. However, in real world applications, sorted pieces tend to be short at length, which will lead to many tiny sorted runs and then end up with poor performance.

B. Natural run

To address the problem mentioned above, we introduce a new concept of natural run, which is defined in definition 1. A natural run is composed of several blocks (items in a block need not to be sorted, while any two items in consecutive blocks are ordered). Figure 1 shows an example of a natural run. In this figure, 3 blocks of items which have no overlap value range are grouped together to be a natural run. A natural run can be treated as a sorted run in the merge phase but the actual sort procedure of each block is delayed until the last block is drained at merging. The concept of natural run exploits much longer pieces that can be ignored in run generation phase, so it can significantly reduce the NVM writes and improve overall performance.

Following this idea, we propose the NVMSorting algorithm leveraging natural runs. We will show that our NVMSorting
can achieve better sorting performance than the MONTRES-NVM algorithm theoretically.

**Definition 1:** (Natural Run) A natural run is a sequence of blocks \( S = \{ b^1, b^2, \ldots, b^n \} \). Each block \( b \) contains identical number of items in value range \([b_{\text{min}}, b_{\text{max}}]\). For \( \forall i, j \) that have \( i < j \), it must hold that \( b_{\text{max}}^i < b_{\text{min}}^j \).

### C. The NVMSorting Algorithm

The basic idea of NVMSorting algorithm is to detect natural runs first and then perform merge sort for the natural runs and normal runs. NVMSorting consists of three phases: (1) natural run detection, (2) run generation, and (3) run merge. Below, we will detail these phases.

1. **Natural Run Detection.** This phase is designed to find the natural runs in the input data. This can avoid loading all input data into DRAM for sorting and then writing them back to NVM. As writes to NVM are much costly, the detection of natural runs help to reduce loading and writing back blocks into NVM. The definition of natural run indicates that an actual run tends to be longer than already sorted pieces used in MONTRES-NVM. So we can infer that natural runs are more common and useful than MONTRES-NVM.

   Differing from the MONTRES algorithm that scans the data to build a Min-Index before generating runs, we record both the maximum and minimum values of each block when scanning the data to build a MinMax-Index. Each element of the MinMax-Index consists of the minimum and maximum value of one data block, and elements are sorted in ascending order. This index is then used to detect natural runs. The algorithm of detecting natural runs is shown in Algorithm 1. Specially, we detect the natural run with the first \( n \) elements as the starting point, and select the longest run as the result.

2. **Run Generation.** In the run generation phase, we also use the merge-on-the-fly mechanism and the run expansion mechanism. Due to the existence of natural run, this phase is different from that of MONTRES. Here, we set block size to \( M \), the size of the whole memory work space of \( N \cdot M \), and divide the DRAM memory into two areas, namely \( W_n \) and \( W_s \), where \( W_n \) is the work space for loading the natural run whose size is \( M \) and \( W_s \) is the work space for loading other data whose size is \( (N-1) \cdot M \). Algorithm 2 shows the process of run generation.

   Before run generation phase, we have scanned the data to get the MinMax-Index and natural run index. The algorithm loads the data into the work space in the order of the smallest elements until the data block in the MinMax-Index is exhausted (line 1). Only one block will be loaded to \( W_n \) at a time for sorting. When the element is exhausted, the next block will be loaded (line 2). Figure 2 illustrates the process of the merge-on-the-fly mechanism. During the process, \( W_n \), \( W_s \), and all the previously generated runs are involved in the merge process (line 7). After merging, we will expand the current run. Because of the existence of natural run, we will use one block of DRAM memory as natural run work space, which leads to the shorter length of the generated run, while the run expansion will reduce this effect and avoid generating too many runs.

3. **Merging Runs.** In the run merge phase, we will exploit the byte addressable feature of the NVM to merge the runs. Unlike MONTRES and the traditional external sorting algorithm, which require loading a block of data into memory each time, we can load only one value from each run.

   We assume that there are \( k \) generated runs. Due to the existence of natural run, in the merge phase, we use a min-heap containing each run’s minimum element with size of \( k+1 \).
we assume that the total amount of data loaded into DRAM memory at this stage is $N_{nl}$, resulting in a read-write cost of $N_{nl} \cdot r$. Due to the existence of merge on-the-fly mechanism, during the run generation phase, we will also read the runs that have been written back before and the qualified data will be written back to the final result. The data that participates in the merge process in the natural run and the qualified data will also be written back to NVM. We assume that the additional reading operation generated in this process is $P_r$, resulting in an additional write operation of $P_w$. For the convenience of later calculation, $P_w$ is divided into two parts: The first part is created by writing back natural run data, set to $P_{w1}$; the second part is created by writing back other data, set to $P_{w2}$. $P_w = P_{w1} + P_{w2}$. The resulting read and write cost is $(P_w \cdot w + P_r \cdot r)$. To sum up, we can get the total cost of reading and writing in the run generation phase by Eq.1:

$$C_{rg} = N \cdot r + (N - N_{nl}) \cdot (r + w) + N_{nl} \cdot r + P_w \cdot w + P_r \cdot r$$

$$= (2 \cdot N + N_{nl} + P_r - N_n) \cdot r + (N + P_w - N_n) \cdot w$$

$$= ((2 + \lambda) \cdot N + N_{nl} + \lambda \cdot P_r + P_w + (1 + \lambda) \cdot N_n) \cdot r$$

(1)

During the run merge phase, all the data not loaded in the natural run will be loaded into DRAM for merging and then written back to NVM. The resulting read-write cost is $(N_n - N_{nl}) \cdot (r + w)$. The elements in the natural run that have been loaded into DRAM memory but have not been written back in the process of run generation also need to be written back to NVM. The total number of data in this part is $N_{nl} - P_{w1}$. Thus, the read and write cost of this part is $(N_{nl} - P_{w1}) \cdot w$. At the same time, the remaining elements in the generated runs will also be loaded into DRAM memory for merging and then written back to NVM. The total number of remaining elements is $N - N_n - N_s - P_{w2}$. Therefore, the read-write cost of this part is $(N - N_n - N_s - P_{w2}) \cdot (r + w)$. To sum up, we can get the total read-write cost in the run merge phase by Eq.2.

$$C_{rm} = (N_n - N_{nl}) \cdot (r + w) + (N_{nl} - P_{w1}) \cdot w + (N - N_n - N_s - P_{w2}) \cdot (r + w)$$

$$= (N - N_{nl} - N_s - P_{w2}) \cdot r + (N - P_{w1} - N_s - P_{w2}) \cdot w$$

$$= ((1 + \lambda) \cdot N - N_{nl} - (1 + \lambda) \cdot N_s - P_{w2} - \lambda \cdot P_w) \cdot r$$

(2)

Based on Eq. 1 and Eq. 2, we can get the total cost of the algorithm by Eq.3. In Eq. 3, $P_{w2}$ approximately equals $P_r$. Eq. 3 shows that the cost of NVMSorting is associated with the size of natural run and the data that can be written into the result directly.

$$C_r = C_{rg} + C_{rm}$$

$$= ((3 + 2 \cdot \lambda) \cdot N - (1 + \lambda) \cdot (N_n + N_s) - P_{w2} + P_r) \cdot r$$

(3)
For the traditional external merge sort algorithm, it is easy to get the total read-write cost by Eq. 4.

\[
C_e = 2 \cdot N \cdot (r + w) \\
= (2 + 2 \cdot \lambda) \cdot N \cdot r
\]

Therefore, we can get the cost reduction of our NVMSorting compared with traditional external merge sort by Eq. 5.

\[
C_{dec} = C_e - C_m \\
= ((1 + \lambda) \cdot (N_n + N_s) + P_{w2} - P_r - N) \cdot r
\]

Eq. 5 shows that NVMSorting has lower cost than the external sort when the value of \( N_n \) and \( N_s \) is larger, which is consistent with the experimental results that will be discussed in Section IV.

IV. PERFORMANCE EVALUATION

In this section, we report the experimental results of NVMSorting. As sorting algorithms are fundamental in computer science and there are a number of existing sorting algorithms, we will compare NVMSorting with several representatives of sorting algorithms. Below, we first introduce the experimental settings in Section IV-A, then we present the results in Section IV-B.

A. Settings

So far, there is one industrial NVM module supplied by Intel in 2019, which is called the Intel Optane DC Persistent Memory [10]. However, in this paper, we still use a simulation way to simulate the hybrid memory using DRAM. There are two reasons for the simulation. First, it is hard to use the Intel DC Persistent Memory to construct various kinds of hybrid memory architecture. Thus, we will not be able to conduct experiments on different configurations of DRAM and NVM. Although it is possible to build multiple servers with different NVM and DRAM capacities, it is too costly because of the high price of the Intel Optane DC Persistent Memory. Second, we mainly focus on the count of NVM writing operations in the experiments. Such a metric can be measured correctly in the simulation environment. In other words, the count of NVM writes of a sorting algorithm will not be impacted by the underlying hardware.

In order to simulate the hybrid memory, we use the same method as [4] to simulate NVM. In particular, We insert delays after cacheline reads and writes. In the experiment, we add 20ns latency for a cacheline read operation and 500ns for a cacheline write to simulate NVM reads and writes.

All algorithms are run on a PC with an Intel CPU i5 8265U@1.6GHz. The CPU has 6MB of L3 cache associated with 12-way groups, and each core has 256KB of L2 cache associated with 4-way groups. The cacheline size is 64 bytes. The memory device is 4 GB LPDDR Samsung memory. We use C++ on Ubuntu 18.04 to implement all sorting algorithms.

The source code was compiled using g++ version 8.3.0 with the -O3 optimization.

We use two datasets, including an unsorted random dataset (denoted as random in the results) and a partially sorted dataset (denoted as partial). In general, we expect that our NVMSorting algorithm will perform better on the partial dataset than on the random dataset.

The performance metrics include execution time \( t \) and the number of NVM writes \( w \). Also, We compare NVMSorting it with six existing sorting algorithms, including quick sort, heap sort, external merge sort, hybrid sort [4], segment sort [4], and MONTRES [8]. And we focus on the comparison of execution time and the number of NVM writes.

B. Results

In the experiment, we set the available DRAM memory size to 10% of the total size of the data, and six sorting algorithms are compared with NVMSorting.

Figure 3 shows the execution time of all sorting algorithms, where MNS, MS, HyS, ExS, SegS, HS, and QS represent NVMSorting, MONTRES, Hybrid sort, External sort, Segment sort, Heap sort, and Quick sort, respectively. We can see that the time performance of NVMSorting is far better than that of SegS, HS and QS in both case. Table III shows the time performance improvement ratio of the NVMSorting algorithm compared with other algorithms. It can be seen that in the case of completely random data, our algorithm has little improvement in time performance compared with MONTRES, even weaker than Hybrid sort and External sort. While in the case of partially sorted data, the time performance
improvement of the NVMSorting algorithm compared with the MONTRES and the External sort is relatively obvious, with the improvement ratio reaching 6.2% and 13.0% respectively.

Figure 3 shows the number of NVM writes for all sorting algorithms. We can see that NVMSorting has the least number of NVM writes. Table IV shows the reduction ratio of NVM writes of the NVMSorting algorithm compared with other algorithms. As shown in Table IV, for the completely random data, NVMSorting has slight improvement in terms of NVM writes, but when running on the partially sorted data, NVMSorting has reduced 27.1% more NVM writes than Hybrid sort and External sort, and 5.7% more NVM writes than MONTRES. This indicates that NVMSorting is particularly suitable for partially sorted data.

We can see from the experimental results that when running on the partially sorted dataset, NVMSorting achieves significant improvement over other sorting algorithms in terms of time performance and NVM writes. When the dataset is completely randomly unsorted, NVMSorting has comparable performance with MONTRES, hybrid sort, and external sort. This implies that NVMSorting is more efficient for partially sorted datasets. For partially sorted datasets, there is high probability of the occurrence of natural run, meaning that \( N_n \) in Eq. 3 is large. At the same time, there are a large proportion of elements in one data block that are less than the minimum value in the next data block. Thus, there will be more elements that can be written into the final result directly, i.e., \( N_n \) in Eq. 3 is large. When the data is completely random, the probability of the occurrence of natural run becomes low, and the merge-on-the-fly mechanism does not work effectively, resulting in little performance improvement. To sum up, NVMSorting is more suitable for partially sorted datasets.

### V. Conclusions

In this paper, we studied the optimization of sorting algorithms for NVM-based hybrid memory architecture and presented a new NVM-friendly sorting algorithm called NVMSorting. NVMSorting is motivated by the MONTRES algorithm that was designed for flash memory. Differing from the original MONTRES algorithm, NVMSorting proposed a new technique called natural run. We developed efficient algorithms for detecting the natural runs in a dataset and sorting data items according to natural runs. We theoretically analyzed the sorting cost of NVMSorting and demonstrated its superiority over the external sorting algorithm. Finally, we verified the performance of NVMSorting on two kinds of datasets and compared NVMSorting to six existing sorting algorithms. The experimental results showed that NVMSorting had higher time performance and fewer NVM writes than MONTRES and other sorting algorithms. In particular, it achieved better performance when running on the partially sorted dataset than on the randomly unsorted dataset.

In the future, we will integrate NVMSorting into database join algorithms [11] to develop efficient sort-join algorithms for NVM-based DBMSs.

### Acknowledgements

This work was supported by the National Science Foundation of China (No. 62072419). Peiquan Jin is the corresponding author.

### References


<table>
<thead>
<tr>
<th></th>
<th>MS</th>
<th>HyS</th>
<th>ExS</th>
<th>SegS</th>
<th>HS</th>
<th>QS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>0.1%</td>
<td>0.3%</td>
<td>0.4%</td>
<td>23.4%</td>
<td>60.4%</td>
<td>91.9%</td>
</tr>
<tr>
<td>Partially sorted</td>
<td>5.7%</td>
<td>27.1%</td>
<td>27.1%</td>
<td>43.9%</td>
<td>74.1%</td>
<td>94.3%</td>
</tr>
</tbody>
</table>

Fig. 4. NVM writes (MNS is our NVMSorting algorithm)
HHML: A Hierarchical Hybrid Modeling Language for Mode-based Periodic Controllers

Zhiming Hu¹, Zheng Wang²,³, Hongjian Jiang¹, Yuyuan Zhang² and Yongxin Zhao¹*
¹ Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
² Beijing Sunwiseinfo Technology Ltd, China
³ Beijing Institute of Control Engineering, China

Abstract—In cyber-physical systems, the controllers are widely designed into mode-based periodic modules, which are used to control physical plants. Such a system can be modeled as a hybrid one, i.e., one of the real-time controller programs and interactive continuous plants that obey dynamical laws. In this work, to facilitate the modeling and analysis of periodic hybrid control systems in the field of aerospace and smart cities, a hierarchical hybrid modeling language (HHML) is proposed, which contains a two-hierarchy structure, i.e., mode-hierarchy and module-hierarchy. The former supports modeling a hybrid system at the abstraction level, while the latter is used to describe the behavior of the modules. The operational semantics is investigated for formal analysis, and the translation rules to hybrid automaton are explored for formal verification. A case study is conducted with the lunar lander to demonstrate the effectiveness of the approach.

Index Terms—Hybrid systems, Cyber-physical system, Formal semantics, Model verification

I. INTRODUCTION

A hybrid system [1] is an interactive system of real-time controllers programs and continuous plants that obey dynamical laws. Such systems are pervasively applied in aerospace, smart cities, and automotive industry, etc. In cyber-physical systems [2], the embedded software and its operating environment have the characteristics of high complexity, uncertainty and high real-time requirements. In general, the controllers are designed into mode-based periodic modules to monitor and control the evolution of physical plants. Formal analysis for such mode-based periodic controllers is still an enormous challenge due to very complicated combinations of computation and control, and high safety requirements of system designs.

Recently, there are a number of formal methods developed for hybrid systems, which can be divided into three main categories: automata [3], process algebra [4] and state diagrams [5]. In automata model, each state contains a large number of differential equations, invariants, and transitions with reset operations. However, the automata model cannot achieve good scalability and composability. In cyber-physical systems, however, a practical model may have hundreds of controllers and sensors. Therefore, automata may not be the suitable choice. As a theoretical basis of formal analysis, process algebra is difficult to be accepted by a wide range of practitioners due its complicated symbols and mathematical logic. The lack of good readability causes even simple controllers to require the help and guidance from experts in the formal field. Hence, the method based on process algebra still needs more improvements. State diagram is a commonly modeling method in the industry of which the typical representative is Simulink/Stateflow. In general, it has the characteristics of a high-level programming language. With the support of tools, the modeling, simulation, verification of hybrid systems can be effectively achieved. However, there are few related researches on its formal semantics.

In this paper, we are motivated by the above methods to put forward a hierarchical hybrid modeling language HHML for the mode-based periodic controllers, which contains mode-hierarchy and module-hierarchy to make the model more scalable and composable. Mode-hierarchy supports modeling a hybrid system at the abstraction level and facilitates the graphical representation of the model that is easy to understand. Module-hierarchy is used to describe the behaviors of the modules, which contains the time predicates unique to the period model. The operational semantics is investigated for formal analysis, and the translation rules to hybrid automaton are explored for formal verification. The contributions of this work are the followings:

• Hierarchical hybrid modeling language. The language provides hierarchy model to support the hybrid systems. The discrete mode has periodicity and supports nesting to describe the control system. The continuous mode represents the physical world by ordinary differential equations. Furthermore, the operational semantics in the mode-hierarchy and module-hierarchy are explored respectively, which helps developers to understand and ensures the correctness of the model built.

• End-to-end translation. Several translation rules of transformation from HHML models to hybrid automata are provided to support property verification in tool Flow*. An illustrative example of a lunar lander is used to demonstrate the feasibility of translation rules.

The rest of the paper is organised as follows. Related work of hybrid system modeling and verification is introduced in Section II. A modeling language HHML is proposed in Section III. In Section IV, the operational semantics is presented. The hybrid automatic translation rules are given in Section V. An example of lunar lander is shown in Section VI. Section VII concludes the paper and introduces the future work.

*Corresponding author: yxzhao@sei.ecnu.edu.cn
DOI reference number: 10.18293/SEKE2021-149
II. RELATED WORK

This section mainly gives some brief introduction to typical hybrid system modeling methods and common verification tools, which provides a reference for this work.

A. Hybrid system modeling methods

Formal methods to model hybrid systems are in progress for many years. The two which have most influenced our language are hybrid action system [6] and Zélus [7].

Hybrid action system was proposed by Mauno et al. It maps continuous-time events to model variables in the form of data flows, so that the overall system behavior can be split into independent differential equations. However, hybrid action system does not support the stepped refinement development methods in action systems, which limits the types of modeling systems. Our discrete mode is hierarchical, so it is suitable for modeling large-scale hybrid systems.

Zélus was a synchronous language with ordinary differential equations proposed by Benveniste et al. It reuses the principles and compilation techniques developed for synchronous languages, extending them to deal with models that mix discrete and continuous-time. However, it uses the type systems to separate discrete and continuous calculations, while we use a clearer discrete mode and continuous mode to distinguish, which makes the interaction between the controller and physical world easier. On the other hand, our language is aimed at periodic controllers, and each discrete mode can have periodicity and use periodicity-related predicates.

B. Hybrid system verification tools

There are many tools that can model and verify specific types of hybrid systems nowadays. Traditional tools include d/dt [8], HyTech, etc., and newer tools contain Flow* [9], SpaceEx [10], etc. They mainly use hybrid automata as the underlying semantic model of hybrid systems. In order to enable the proposed hybrid modeling method to complete the verification of the property in these verification tools, there are many researches on translating from hybrid modeling languages to hybrid automata. For example, in [11], a subset of Simulink language was proposed to translate into hybrid automata. In [12], the conversion rule from ECML to SpaceEx model was proposed. Its essence is the translation of part of ECML to linear hybrid automata.

The HHML model was translated into the above four tools respectively, with requirements being satisfied potentially, to verify the common examples of hybrid system. Taking the running time, running scale and supported operation format into consideration, Flow* is finally chosen.

III. HYBRID DESCRIPTION LANGUAGE

This section proposes the hybrid description language HHML to provide the rich control logic, and events that can drive the conversion among different modes. HHML is a two-hierarchy structure containing mode-hierarchy and module-hierarchy. The former one supports to model a hybrid system at the abstraction level, while the latter one describes the detailed behavior of modes.

A. Mode-hierarchy syntax

The following syntax elements are in support of the modeling of the hybrid system architecture at an abstract level:

\[
\begin{align*}
HModel &::= (Dictionary, Modes) \\
Dictionary &::= \{ \text{var} | \text{var} = (name, attri, type, initval)\} \\
Modes &::= (dModes, cModes) \\
dMode &::= (name, period, (dflow | dModes), dTrans) \\
cMode &::= (name, cflow, cTrans) \\
dTrans &::= (dn, priority, dguard, dn') \\
cTrans &::= (cm, priority, cguard, cm') \\
dguard &::= \text{cond} | \text{Duration(}\text{cond}, c\text{)} | \text{After(}\text{cond}, c\text{)} \\
cguard &::= \text{When(}\text{cond}\text{)}
\end{align*}
\]

HModel indicates the hybrid model which is composed of Dictionary and Modes. Dictionary is the set of four-tuple variables var, where, name is the label of the variable, attri is the attribute which can be discrete, continuous and constant, type is the basis type such as Boolean, int, float, and initval means the initial value.

Modes illustrates the behavior of the hybrid system which is made up of discrete modes dModes and continuous modes cModes. A discrete mode dMode is used to describe the control system which consists of label name, period, discrete control flow dflow or sub-modes dModes, and discrete transitions dTrans. A continuous mode cMode denotes the changes in the physical world with the differential equations, which consists of name, continuous statements cflow and continuous transitions cTrans.

dTrans is the transfer relationship between discrete modes including the source mode dn, priority, guarded conditions dguard, and target mode dn'. Duration and After are HHML's special time predicates in dguard to express the property of periodicity based on the basic Boolean expression cond and constant c. Duration(\text{cond}, c) is true in a period p if the first c periods within the current period meet \text{cond}. After(\text{cond}, c) is true in a period p if there is another period p' such that \text{cond} can be satisfied in period p' and it travels c periods from p' to p.

cTrans is similar to dTrans, where cguard denotes the continuous guarded condition, cm and cm' denote the source and target continuous mode. cguard uses When(\text{cond}) to mean that the system will always wait for \text{cond} to be satisfied. The difference between dTrans and cTrans is that cTrans is not controlled by the period. Therefore the transition occurs immediately when cguard is met, and time predicates are missing in cguard.

B. Module-hierarchy syntax

Module-hierarchy is divided into discrete flow dflow and continuous flow cflow, which denotes the calculation process of the hybrid model, and the following elements are used to specify the behavior of the hybrid system.

\[
\begin{align*}
\text{dflow} &::= \text{declare} \mid \text{stmts} \mid \text{dflow;dflow} \\
\text{stmts} &::= \text{pststmt} \mid \text{cstmt} \\
\text{pststmt} &::= x := \text{stmt} | x \leftarrow \text{cv} | \text{call func} | \text{skip} \mid \downarrow \\
\text{cstmt} &::= \text{stmt}; \text{stmt} \mid \text{while cond do stmts} \mid \text{if cond then stmts else stmts} \\
\text{cflow} &::= \text{eq} \mid \text{until cond} \\
\text{eq} &::= \text{der v = expr} \mid \text{eq} \parallel \text{eq} \mid \text{Idle}
\end{align*}
\]
...dflow represents the execution task and calculation process of the discrete mode, including local declarations declare, control statements stmts and the combinations of dflo...TABLE I: State Transition Rules at the Mode-Hierarchy

<table>
<thead>
<tr>
<th>Transition Type</th>
<th>Transition Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>dflow(dm) ≠ empty</td>
</tr>
<tr>
<td>Enter_Sub</td>
<td>(cm, dm, Begin, per, Tr) → (cm, dm, Execute, per, Tr')</td>
</tr>
<tr>
<td>Execute</td>
<td>dflow(dm) = empty</td>
</tr>
<tr>
<td>(Enter_Sub)</td>
<td>(cm, dm, Begin, per, Tr) → (cm, dm', Begin, per, Tr)</td>
</tr>
<tr>
<td>Execute</td>
<td>execute(dm, dflo, Tr') = Tr'</td>
</tr>
<tr>
<td>no_dTran</td>
<td>(cm, dm, End, per, Tr) → (cm, dm, Begin, per + 1, Tr)</td>
</tr>
<tr>
<td>dTran</td>
<td>dTran ∈ dTrans . ¬(Tr ⇒ dm.dguard)</td>
</tr>
<tr>
<td>cTran</td>
<td>(cm, dm, l, per, Tr) → (cm', dm, l, per, Tr')</td>
</tr>
</tbody>
</table>

(Execute) describes that when the discrete mode enters the execution phase, the system processes the variables according to the discrete control flow and then changes from the Execute phase to the End phase. execute(dm, dflo, Tr') = Tr' means the result of the execution.

(no_dTran) indicates that when none of the discrete transition conditions can be met, the discrete mode which the system is in remains unchanged, and the phase of the system is updated from End to Begin. Then the period is increased by 1.

(dTran) shows that when at least one discrete transition condition is met, the system will select the highest priority for transition. The current discrete mode dm will transfer to the target discrete mode dm', and the operating period will be reset to 1. Hpri(dm) = 'tran' ∈ dTrans . ¬(dm.dguard ∧ dm.priority > tran.priority) is defined to denote the highest priority for transition.

(cTran) expresses that the continuous mode will only transfer to the target continuous mode cm' with the highest priority Hpri(dm) when the transfer conditions are met. The transition will occur immediately when the eguard is met, because the eguard is always in a wait state.

B. Operational semantics of the module-hierarchy

A triple module_config is used to represent the semantics of the module-hierarchy as shown below.

module_config ::= (σ, stmts, status)

where:

- σ stands for the set of variables.
- stmts represents the statements to be executed.
- status denotes the states of the module layer, and it includes three states: term, wait and div. State term expresses that the previous statement runs successfully and you can continue to execute the current statement. State wait indicates that the previous statement is blocked and a guard event is needed to activate the system. State div means that the system has an error and cannot execute subsequent programs.
The state transition rules of the Module-Hierarchy are given in Table II. 

(Assign) and (Sample) respectively denote the assignment of discrete variables and the sampling of continuous variables. If the current state is \( \text{term} \), the operation can be performed, and the variables in the variable set \( \sigma \) are modified. (Func) means that calling the function \( \text{func} \) which makes the current variable set \( \sigma \) become \( \sigma' \) and sets the currently executed statement to empty. (Skip) means doing nothing. (Div) indicates that the system lies in a divergent state when \( \bot \) holds. 

(Seq) describes the systems executing the sequential statements. (Loop1) and (Loop2) respectively represent that the loop statement holds or not. (Loop1) denotes that the statements are in the loop and the entire loop will be executed as sequential statements, while (Loop2) denotes the termination of loop. 

(Cond1) and (Cond2) are two transition rules to denote the if statement, where the former holds when the condition is true and the latter holds with false condition. 

(EQ1) denotes the condition is meet and the \( \text{until} \) statement is executed. While the condition is unsatisfied, (EQ2) denotes the system must wait for \( \delta \) periods till the condition holds. 

V. TRANSLATIONS OF HHML

This section introduces the modeling essence of \( \text{Flow}^* \) and proposes some translation rules to translate the model established by HHML into a hybrid automaton. According to the generated automaton, formal verification of safety and reachability has been successfully implemented by the tool \( \text{Flow}^* \).

A. Hybrid automata

\( \text{Flow}^* \) [9] works on systems that can be modeled by hybrid automata. Hybrid automata can be expressed as:

\[
\text{(loc, var, inv, flow, trans, guards, resets, init)}
\]

where:
- \( \text{loc} \) is a finite set of continuous states, also called modes.
- \( \text{var} \) consists of several real-valued variables.
- \( \text{inv} \) means the invariant of each mode.
- \( \text{flow} \) represents the continuous dynamics defined by ordinary differential equations of each mode.
- \( \text{trans} \) is the set of possible transition between modes.
- \( \text{guards} \) is the set of transition conditions between modes.
- \( \text{resets} \) assigns a reset map to a jump. After a jump occurs, the values of the continuous variables will be updated according to the reset mapping.
- \( \text{init} \) denotes the initial of the automaton.

B. Translation rules

To simplify the expression of hybrid automata, we use the \( \text{jumps} \) set to denote the union set of \( \text{guards} \), \( \text{resets} \) and \( \text{trans} \). \( \text{jump} \) represents an execution of system resets.

\[
\text{jumps} := \{\text{jump} \mid \text{jump} = \{l_{\text{begin}}, l_{\text{end}}, \text{guard}, \text{reset}\}\}
\]

Therefore, hybrid automata can be expressed as a six-tuple .

\[
(\text{loc}, \text{var}, \text{inv}, \text{flow}, \text{jumps}, \text{init})
\]

Now, the translation rules containing the variables, discrete modes, continuous modes and some flows in module-hierarchy are shown in turn.

1) Variables: Variables \( v \) are translated by the below rule where using \( "-" \) to denote the unchanged elements.

\[
\text{Tr}(v) = (-, \text{var} \cup v, -, -, -) \cup v.inival
\]

The variables in HHML are divided into continuous, discrete and constant. The types include integer, floating-point and Boolean. The variables in \( \text{Flow}^* \) are unified as floating-point continuous state variables. Therefore, the Boolean variables are transformed to 1/0 and other variables are converted into floating-point. Then these variables can be converted into state variables in the hybrid automaton directly. To reduce the number of translated variables, we will change the constants to values. Assigning initial values \( \text{inival} \) to variables in HHML is corresponding to the initial variables \( \text{init} \) in hybrid automata.

Since there is no discrete modes in hybrid automata, the names of discrete modes are added as variables into \( \text{var} \). We use flag 1/0 to distinguish whether the system lies in the discrete mode. Finally, time term \( t \) is added to record the period in automata whose initial value is set to 0.

2) Discrete modes: The discrete mode supports mode nesting in HHML which may contain several sub-modes. So before translating, the discrete mode has to be flattened, i.e.,
all discrete modes after simplification do not contain sub-modes, and keep the semantic consistency during the translation. Flattened discrete modes can be described as below.

\[ dmodes' = \{ dm \mid dm.dflow \neq empty \land dm \in dmodes \} \]

Since \( loc \) is a set of states in the hybrid automata, the discrete modes need to be translated into each state. A certain state \( l \in loc \) is used in the translation rules and other states are the same. The following rule is about when the discrete mode transfers.

\[
Tr(dm) = (-, -, -, flow \cup t = 1, jumps \cup jps, -)
\]
where \( jps = \{ jp \mid jp = (l, t, (t \geq dm.period; dm.name == 1; dguard),(dm.name = 0; dm'.name = 1; dm'.dflow; t = 0)) \land (dm, -, dguard, dm') \in dm.dTrans \} \]

In order to translate \( dm.period \), \( t = 1 \) is added to each \( flow \) to represent the periodic process. Condition \( t \geq dm.period \) is attached to \( guard \), and \( t \) will be set to 0 in \( resets \) to indicate the end of the period. \( dguard \) is converted into \( guards \) to translate the conditions of discrete transfer process. Once the transition occurs, the discrete control flow of the target mode \( df.dflow \) is executed, and the current mode is modified to \( dm' \). There is no transition between continuous modes, hence the target state is still the source state \( l \).

The following rule is about when the discrete mode does not transfer.

\[
Tr(dm) = (-, -, -, flow \cup t = 1, jumps \cup jps, -)
\]
where \( jps = \{ jp \mid jp = (l, t, dm.period; dm.name = 1, dm.dflow; t = 0) \} \}

The rule indicates that when the discrete mode does not transfer, it will execute \( dm.dflow \) and then enter the next period at the end of the period.

3) Continuous modes: Continuous modes are translated by the rule as follows.

\[
Tr(cm) = (loc \cup cm.name, -, inv \cup cm.cond, flow \cup cm.eq, jumps \cup cm.eguards, -)
\]

As mentioned above, the continuous mode is a triple in HHML. The corresponding translation is performed between the hybrid automata and the continuous mode. The continuous modes’ \( name \), differential equation \( eq \) and termination condition \( cond \) will be translated to \( loc, flow \) and \( inv \) in hybrid automata respectively. The transfer between continuous modes is equivalent to the \( jumps \) behavior in the hybrid automaton, while \( resets \) in \( jumps \) does nothing during the transfer.

4) Some flows in module-hierarchy: Since Flow* only supports part of the discrete control flow, conditional statement along with the time predicates \( Duration \) and \( After \) is translated to enable expressive models to be verified.

First, the translation rule of conditional statement \( if \ cond \ then \ stmt1 \ else \ stmt2 \ in \ dflow \) can be expressed as:

\[
Tr(dflow.cd) = (-, -, -, -, jumps \cup jps, -)
\]
where \( jps = (l, t, cond, stmt1) \cup (l, l, -cond, stmt2) \)

\( dflow \) that contains the conditional statement will be split into two, and conditional statements will be replaced with \( stmt1 \) and \( stmt2 \) and set to the \( reset \) respectively. The corresponding \( cond \) and \( -cond \) are added to the \( guards \).

Next, rules of time predicates \( Duration \) and \( After \) necessary for modeling periodic hybrid systems are introduced. They only focus on changing the \( guards \) and \( resets \) in the \( jumps \) behavior.

\[
Tr(dflow.Duration(cond, c)) = (-, var \cup cnt, -,-, jumps \cup jps, init \cup cnt = 0)
\]
where \( jps = (-, -, cond, cnt = cnt + 1) \cup (-,-,-, -cond, cnt = 0) \cup (-,-, cnt \geq c, -) \)

\[
Tr(dflow.After(cond, c)) = (-, var \cup cnt, -,-, jumps \cup jps, init \cup cnt = 0)
\]
where \( jps = (-, -, cond, cnt = cnt + 1) \cup (-,-,-, cnt > 0, cnt = cnt + 1) \cup (-,-, cnt \geq c, -) \)

An additional count variable \( cnt \) is introduced here. For \( Duration \), when a period of the discrete mode ends, \( if \ cond \) is true, \( cnt \) will increase by 1, otherwise it will be reset to 0. When \( cnt \geq c \), the expression of \( Duration(cond, c) \) is true.

Similarly, for \( After(cond, c) \), when \( cond \) is true or \( cnt > 0 \), \( cnt \) will increase by 1 at the end of the period, and when \( cnt \geq c \), the expression is true.

VI. Case study

In this section, the process of lunar lander’s slow descent is modeled into the hybrid system by HHML. Then, the model is translated into a hybrid automaton and Flow* carries out the reachability analysis towards it.

A. Model

The model analyzed in this case study is taken from the descent guidance control program of a lunar lander in [13]. In brief, it is a sampled data control system composed of physical devices and control programs. The thrust exerted on the lander is constantly adjusted by the system to ensure that the lander remains stable during the slow descent phase. So that it enters the free fall phase smoothly and completes the landing. For the specific meaning and value of each parameter, please refer to [13]. This paper only models from the parameter level.

The hybrid system is divided into the current stage of the guidance program and the lander dynamics. The guidance program (i.e. discrete mode) modeled by HHML is shown in top half of Figure 1. The slow descent phase of the guidance program is executed periodically in a sampling period. At each sampling point, various sensors will sample the current state of the lander. The sampled values will be calculated in the guidance program, and the control command will be output, which will then affect the dynamics of the lander. When more than 10 seconds have passed during the slow descent phase (approximately 80 periods), and the height of the lander is less than 6 meters, the system will switch from the slow descent phase to the free fall phase and send out the signal.

Furthermore, the lander dynamic (i.e. continuous mode) modeled by HHML is shown in bottom half of Figure 1 which is considered only in the vertical direction. \( dynamic_1 \) and \( dynamic_2 \) indicate the change of the lander under different
thrusts. After receiving the signal to change to free fall, the dynamics will change to dynamic_3 to indicate free fall.

**Fig. 1: Lunar Lander Modeled in HHML**

**B. Translation and verification**

The translated model shown as Fig. 2 will be verified with regard to three properties [13] in the following.

First, the speed fluctuation of model is supposed to satisfy $|v - v_{lsu}| < 0.05$. Then, the lander will eventually reach the surface of the moon. Finally, the speed of the lander should not exceed $v_{Max}$ when arriving at the destination.

**Fig. 2: Translated Mode in Hybrid Automata**

The computation costs 9 minutes on the platform with 2.4 GHz Intel Core i5 CPU and 16GB RAM running macOS. The reachable sets of the translated model are given in Figure 3.

**VII. CONCLUSION AND FUTURE WORK**

This paper has introduced a hierarchical hybrid modeling language (HHML) for periodic controllers. The language uses periodic and hierarchical discrete modes to formalize the control system, and continuous modes to model the physical environment. In addition, translation rules help translate the model into hybrid automata, and implement verification of the properties on the verification tool Flow*. The verification results show that the lander can finally reach the lunar surface smoothly and safely.

In the future, we plan to apply HHML to more cases in smart cities, and support more verification tools.

**ACKNOWLEDGEMENTS**

This work is supported by National Key Research and Development Program (2019YFB2102600), Lab of High Confidence Embedded Software Engineering Technology, Beijing Institute of Control Engineering, No: HCESET-2019-1.

**REFERENCES**


Patterns for Reuse in Production Systems Engineering

Kristof Meixner∗†, Arndt Lüder‡∥, Jan Herzog§, Dietmar Winkler∗†, Stefan Biffl†∥

∗Christian Doppler Laboratory SQI, †Inst. of Information Sys. Eng., TU Wien and ‡CDP, Austria
E-Mail: [first],[last]@tuwien.ac.at
‡Inst. of Ergonomics, Manufacturing Sys. and Automation, Otto-von-Guericke U., Germany
E-Mail: arndt.lueder@ovgu.de
§Department PPG-M/D, Volkswagen AG, Wolfsburg, Germany
E-Mail: [first],[last]@volkswagen.de

Abstract—In Production Systems Engineering (PSE), domain experts aim at reusing production processes implemented as Industry 4.0 assets and software. However, the knowledge on reusable assets is often scattered on multi-disciplinary engineering artifacts and domain experts, making it hard to find suitable reusable assets and map them to requirements. In this paper, we (i) identify challenges and requirements for reuse in PSE based on a domain analysis; (ii) introduce the Industry 4.0 Asset Network (I4AN) that integrates multi-disciplinary dependencies between the assets and exposes recurring patterns; and (iii) present four patterns for reuse in PSE that aim at improving reuse efficiency and risk. We evaluate the I4AN with reuse scenarios in a feasibility study. The study results indicate that the I4AN model satisfies the elicited requirements and enables PSE domain experts to identify patterns for reuse in their contexts.

Keywords—Reuse, Production Systems Engineering, Industry 4.0 asset, Industry 4.0 component.

I. INTRODUCTION

The Industry 4.0 (I4.0) initiative1 has led to an increased focus on research related to Production Systems Engineering (PSE) in various research fields [1]. The I4.0 initiative envisions flexible and highly customizable production systems that interconnect modern manufacturing with the latest information and communication technology, so-called Cyber-Physical Production Systems (CPPSs) [2] that can self-adapt to particular conditions. These CPPSs incorporate I4.0 assets representing objects of perceived or actual value, such as products, processes, or resources [3]. The Asset Administration Shell (AAS), their standardized digital representation [3], can describe their skills [4] and adapt the I4.0 assets to changes in the production environment. The aim is to fulfill business demands for increased flexibility and distribution of production, i.e., production as a service, and to react to shorter product life-cycles with reduced PSE project duration and effort [5].

These demands require the (partial) reuse of process and resource solutions from previous projects or standardized catalogues [6], [7]. Examples in automotive manufacturing are position and screw tasks, like screwing a dashboard into a car. In such cases, product type variants and their parameters vary, e.g., where and how tightly to screw which kind of screw to a dashboard. Yet, the processes and production resources executing these tasks, like robot arms, are quite similar. In addition, parts of the software controlling the resources and orchestrating the overall production system can be reused.

Reuse in PSE depends on efficiently identifying recurring patterns that can be integrated into a production system. These patterns need to follow reference architectures [8], [9] of (i) product types, e.g., car types, (ii) production processes, e.g., screwing processes, and (iii) production resource types and instances, e.g., screwing robots. Reuse also requires a pattern description on type and instance levels to facilitate referring to vendor catalogues or previous projects [6], [10].

The engineering of a production system is a collaborative effort of experts coming from many disciplines, like mechanical, electrical, and software engineering [11]. However, traditionally much of the engineering information is hidden in scattered engineering artifacts and much of the knowledge is implicit domain knowledge of engineering experts [5] (cf. Section IV). Furthermore, there is insufficient interdisciplinary exchange between the domains, leading to hard to extract/collct/validate dependencies from heterogeneous engineering artifacts and domain experts [12]. Hence, it is crucial in this multi-disciplinary environment to thoroughly model the (interdisciplinary) dependencies and boundaries in pattern analysis to reduce the risk of broken reusable assets.

Hence, we raise the main research question: What approach can PSE experts use to efficiently identify patterns from existing engineering knowledge for reusing Industry 4.0 assets and related artifacts?

In this paper, we (i) identify challenges and requirements for knowledge reuse in PSE, (ii) introduce the Industry 4.0 Asset Network (I4AN), a model to integrate the scattered knowledge and enable engineers to identify patterns for reuse to improve the effectiveness and efficiency of the PSE life-cycle, and (iii) present four recurring high-level patterns in PSE as a basis for identifying applied solution patterns for similar problems. We evaluate these contributions with an instance of an I4AN. Therefore, we investigate to what extent typical reuse scenario questions can be answered as queries to the I4AN.

The remainder of this paper is structured as follows: Sec-

DOI: 10.18293/SEKE2021-150

1 Industry 4.0 Initiative: https://www.plattform-i40.de
Section II summarizes related work. Section III presents the research questions and method. Section IV introduces an illustrative use case and identifies requirements for reuse in PSE. Section V introduces the Industry 4.0 Asset Network (I4AN) for reuse in PSE and four high-level reusable patterns. Section VI reports on a feasibility study to evaluate the I4AN capabilities and discusses the results and limitations of the research. Section VII concludes and outlines future work.

II. RELATED WORK

This section summarizes related work on Production Systems Engineering (PSE), knowledge management, and reuse.

A. Production Systems Engineering

PSE is a multi-disciplinary process that involves various disciplines, like mechanical, electrical, and software engineering [11]. Engineering teams iteratively perform tasks, like mechanical design or implementation of the control software [13], to engineer the desired production system.

In PSE, engineers create various types of engineering artifacts and models [5], [14]. However, the used formats and tools have traditionally been optimized for a single discipline, and while engineers are well connected within their domains, there is often an insufficient interdisciplinary exchange. Further, the engineering artifacts and information are scattered throughout the engineering landscape [5]. Much of the engineering knowledge is implicit knowledge of the domain experts. These issues pose an increasing challenge related to information management and reuse within PSE projects [15].

Yet, for the suitable and correct production system design, it is crucial to exchange information and knowledge between the disciplines effectively and efficiently [16]. In addition, the reuse of designs and specifications for recurring problems, e.g., using a robot type for similar tasks, improves the quality and helps to reduce PSE project duration, effort, and risk [8].

In this paper, we introduce a network, based on explicitly linked assets and artifacts, to provide the foundation for domain experts to link their specific knowledge representations.

Industry 4.0 (I4.0) addresses the overall digitalization and networking of production system elements, i.e., I4.0 assets, towards Cyber-Physical Production Systems (CPPSs). I4.0 assets are physical or immaterial objects of perceived or actual value [3]. An increasing focus can be recognized on product, process, and resource-related I4.0 assets, which we mainly refer to. An Asset Administration Shell (AAS) provides a digital representation [3] of I4.0 assets with their property views and skills. These descriptions should include the information and knowledge for an automated orchestration, which requires explicit knowledge on the I4.0 assets and their dependencies.

Pfrommer et al. [17] define a skill as the ability of a resource to perform a process, while a production skill gives the requirements [4]. Candidio et al. [18] understand a skill as ability to perform actions that are needed to support the production process. Meixner et al. [10] described how to abstract skills of resources from process requirements. Hence, models must represent the required skills of processes and the provided skills of resources [17], [19]. However, the identification of reusable I4.0 Asset candidates requires representing skills as I4.0 Assets to provide the abstraction for the digital representation of boundaries between reusable patterns.

In this paper, we represent skills for the first time as I4.0 Asset, as an abstraction between processes and resources to foster the identification of reusable I4.0 Asset candidates.

B. Modeling Engineering Knowledge in PSE

Sabou et al. [20] introduced a knowledge graph for reuse in the software engineering domain, but without multi-model links that facilitate reuse in PSE.

For modeling Cyber-Physical Production Systems (CPPSs), two main research and development directions have been pursued. First, IT systems engineering uses CASE tools based on UML [21]. As a result, systems engineering methodologies have been created utilizing domain-crossing modeling standards like SysML [22]. Second, engineering data exchange preserves the multi-model nature of PSE knowledge and builds on standardized data formats like AutomationML [16] to make data integration more efficient. In this paper, we build on cross-linking assets and engineering artifacts as a basis for an improved reuse considering dependencies.

Both directions require for explicitly modeling PSE knowledge to reflect the specifics of this domain, including (i) modeling part-whole relations, (ii) connections between components [23], and (iii) technical dependencies of the various involved technical disciplines [24] The authors observed in PSE containment hierarchies to be well-established and frequently used to organize assets in PSE models. Furthermore, discipline-specific dependencies are often represented in discipline-specific models as interfaces.

Feldmann et al. [25] introduced an approach for managing inconsistencies in a multi-disciplinary multi-model environment using links between objects in PSE. However, the approach by Feldmann et al. [25] does not consider I4.0 assets and skills as first-class citizens in PSE. However, this integration is a foundation for better identifying reusable assets based on the digital representation of their skills. In this paper, we build on their meta-model [25] to integrate links between I4.0 assets coming from several engineering disciplines.

C. Reuse in Production Systems Engineering

Main approaches to reuse are (i) clone and own [26] and (ii) reuse of components, such as software libraries. However, these general reuse approaches do not sufficiently cover requirements in multi-disciplinary environments, like PSE.

In PSE, several reference frameworks address the reuse of assets. The guideline VDI 2206 [27] describes the V-Model as a procedure for structured PSE. It encourages to reuse requirements and partial implementations in later phases, like the test phase, without mentioning how. Jazdi et al. [8] provided first methodologies related to the systematic identification of

\footnotetext[2]{SysML: https://www.sysml.org}

\footnotetext[3]{AutomationML: https://www.automationml.org}
reusable system components. Most of them are based on the idea of mechatronic systems [28] following the VDI 2206 [27].

The guideline VDI 3695 [29] understands reuse as a method for engineering optimization and defines five types of reuse, i.e., Reuse Levels (RL): Reuse (i) by employees on their own accord (RL-A), (ii) controlled within the project (RL-B), (iii) controlled from a central point across all projects (RL-C), (iv) based on a reference model (RL-D), and (v) based on internal and external standards (RL-E). The effectiveness and efficiency of reuse of assets depends on the level of reuse maturity [29] and the relations between assets [9]. Yet, this information is insufficiently available in PSE due to scattered artifacts and information.

In software engineering, one specific domain in the PSE engineering process, design patterns [30] are a widely adopted standard for reuse. Design patterns aim at developing software faster and in better quality while reducing risks and cost [30]. Therefore, design patterns provide adaptable design solution templates to general problems that software developers face. A design pattern consists of (i) a pattern name, (ii) a description of a problem that should be solved, (iii) a solution description with its elements and dependencies, and (iv) implications of the pattern, such as benefits and limitations. Software design patterns can serve as a blueprint for PSE design patterns but need to be adapted to the multi-disciplinary context of PSE.

In this paper, we build on the guideline VDI 3695 to describe the reuse maturity in PSE and the idea of mechatronic units as reusable entities [28]. Furthermore, we build on design patterns as a concept to identify reusable patterns in PSE.

III. RESEARCH QUESTIONS AND APPROACH

In this paper, we follow the Design Science methodology [31] to investigate how to improve identifying I4.0 assets for reuse in PSE. Therefore, we (i) conducted a domain analysis in the automotive industry, (ii) condensed a representative use case, and (iii) elicited requirements on I4.0 asset reuse with domain experts at medium-to-large European PSE companies (cf. Section IV).

Considering identified gaps in the related work and requirements in PSE, we formulate the following research questions.

**RQ1a. What model and elements facilitate identifying I4.0 assets for reuse in PSE?** The systematic reuse of I4.0 assets in engineering fosters quality and efficiency [8]. However, in PSE, the knowledge required for reuse often consists of heterogeneous information and implicit knowledge, scattered across the engineering landscape. To address RQ1a, we investigated recurring engineering artifacts from the domain analysis to identify knowledge elements that help engineers in efficiently identifying reusable assets. Our contribution is the Industry 4.0 Asset Network (I4AN) as a foundation to explore assets suitable for reuse.

**RQ1b. What connections between system parts and engineering artifacts represent dependencies in an I4.0 asset network as a foundation for identifying sets of reusable assets?** Connections and relationships between I4.0 assets provide data to understand internal and external dependencies of CPPS assets. These dependencies are crucial to coherently identify and explain which potentially reusable assets can be reused as-is or require further assets to be included to correctly reuse them. To address RQ1b, we build on the Industry 4.0 Asset Network (I4AN) , coming from RQ1a, and investigated which links represent internal and external dependencies that are relevant to facilitate the reuse of assets. Our contributions focus on the classification of dependencies in the I4AN that are crucial to identify sets of reusable assets.

**RQ2. Which basic patterns for reuse facilitate identifying best-practice pattern candidates for PSE?** For identifying patterns for reuse, engineers require a starting point in their particular context. For instance, engineers are likely to recognize a pattern as an initial set of assets and their dependencies to other assets. Basic patterns, which occur independently from the particular context of the PSE project, can represent such a starting point. From domain analysis and discussions with engineers, we identify basic patterns that regularly occur in PSE. These patterns provide blueprints to help engineers identify reusable assets in I4AN instances.

Each research question addresses parts of the overarching question (cf. Section I) tying together model elements, their dependencies, and patterns for the efficient reuse of CPPS knowledge. We evaluate the I4AN in a feasibility study for the use case “Car Body with Screwed-on Parts” (cf. Section IV). Therefore, we use data from a sample of artifacts from the domain analysis. We investigate to what extent advanced reuse scenario questions can be answered by queries to the I4AN.

IV. ILLUSTRATIVE USE CASE

This section introduces the use case “Car Body with Screwed-on Parts”. We condensed the use case from a domain analysis in the automotive manufacturing domain. The analysis was conducted in a setting with 80 types of screwing robot cells and 27 robot types.

In automotive manufacturing, human workers collaborate with industrial robots in mounting lines to place and screw various components onto a car body using screwdrivers. Typical mid-class cars contain screws of 80 screw types at 1,500 to 1,800 screw positions. Figure 1 shows the use case with its I4.0 assets and their connections. The left-hand side shows a screwing process consisting of two steps: (i) positioning the dashboard and the screws and (ii) fastening the screws. Both steps are characterized by process requirements, defining the necessary skills of the resources including technical or economic parameters. In PSE, relevant resources, i.e., resource hierarchies, (see right-hand side of Figure 1) are selected and orchestrated to provide the required skills [17], [19].

In theory, one can engineer an optimized robot-screwdriver combination for each screw type to maximize production effectiveness and efficiency. Yet, this approach might lead to around 80 different robot and screwdriver types, adding significant costs for installation, maintenance, and expert knowledge.

In practice, PSE aims at cost-optimized system designs [32]. Hence, a sufficiently effective and efficient robot-screwdriver combination to each screw type can be assigned, minimizing
the number of robot types and investments in spare parts and know-how. This approach may significantly reduce costs in comparison to a high-variety approach.

Identifying an optimized set of robot-screwdriver combinations for the high number of different screws types and positions requires the identification of (i) resources that can execute several varying screwing tasks and (ii) engineering artifacts that can be reused, such as control programs. However, identifying suitable solutions is difficult due to the scattered information and the engineers’ implicit knowledge.

Achieving these advantages requires a set of reusable patterns. This set of patterns can be completed by (i) identifying similar components within existing engineering projects, (ii) mapping these components to expected future requirements, and (iii) abstracting these components with respect to possible adaptations for application-case related parameterization [8].

Requirements for I40 Asset Reuse. From the domain analysis, we elicited the following requirements Rx towards asset reuse in PSE with eight domain experts from five medium-to-large European PSE companies.

R1. I40 Asset Map. Domain experts require an Industry 4.0 Asset Map, i.e., an overview on the assets in the planning phase to explicitly represent implicit knowledge and relevant information as a context for reuse, currently scattered across various engineering artifacts. This requirement is adapted from software engineering, i.e., documenting the project structure and software artifacts, to multi-disciplinary PSE assets in the Product-Process-Resource (PPR) scope.

R2. I40 Dependency Network. As a basis to identify patterns for reuse in PSE, domain experts require an explicit representation of the links and dependencies of and between assets coming from several engineering disciplines. This concerns mainly three different views. Product engineering requires links between product components, processes, and their required skills that a process requires from a resource to automate the process. Systems engineering concerns relationships between resources and their provided capabilities. The assets and dependencies can be represented in an Industry 4.0 Dependency Network that adds information and knowledge required for reuse to the Industry 4.0 Asset Map.

R3. System Boundary. For reuse, a system or subsystem containing the reusable assets needs to have a clearly defined boundary. System boundaries are a means to group assets into a meaningful set of assets that can be reused. A boundary also allows to investigate incoming and outgoing dependencies.

The experts rated their company’s maturity level of asset reuse, using the VDI 3695 classification, at reuse levels RL-C (controlled from a central point) or RL-E (reuse based on internal and external standards).
Thus, system boundaries serve as a basis for systematically reusing (parts of) a solution that was used in previous projects. Without a clear boundary, it is unclear which elements can, should, or have to be included in a set of reusable assets. Furthermore, system boundaries enable developing and using metrics, like complexity, to compare patterns.

**R4. Solution Design Abstraction.** As a foundation to identify reusable patterns, domain experts need a representation of solution design candidates at a suitable level of abstraction. This abstraction is required to allow the adaptability and portability of a pattern to similar problems with varying characteristics. For example, to make a solution for a position task reusable requires hiding unnecessary attributes and dependencies. In the use case, the robot positioning accuracy is a relevant characteristic, while the way how the robot moves might be irrelevant. Solution Design Abstraction facilitates (i) generalizing from a particular solution instance to a more general level of problems and (ii) finding reusable solution candidates in similar or historic designs.

The following section builds on this use case to illustrate a novel knowledge representation model for Industry 4.0 Assets for identifying patterns for reuse.

**V. PATTERNS FOR REUSE IN PSE**

This section presents the Industry 4.0 Asset Network and four basic patterns to identify concrete patterns for reuse.

**A. I40 Asset based Network with Dependencies**

To address RQ1a and RQ1b, we investigated the data of robot cells with up to two robots from the use case context with domain experts. From this data, we determined knowledge elements that we can use for identifying abstract patterns for reuse. These elements were used to build a condensed metamodel as the foundation for the I4AN. This section illustrates the metamodel and the I4AN using the car body with screwed-on parts use case from Section IV.

![Diagram of Asset and Link meta model](https://example.com/diagram.png)

**Fig. 2. Asset, (Engineering) Artifact and Link meta model, based on [33]**

Figure 2 shows the metamodel (in UML notation) containing the Asset class, one of the Industry 4.0 Asset types product, process, resource, or skill. An Asset can be a specialization (isa relation, e.g., an electric screwdriver is a type of screwdriver) and/or a part (is-part-of relation, e.g., a bit is part of a screwdriver) of another asset. An Artifact is an engineering object created during design time, e.g., an electrical plan or robot program, or during runtime, e.g., a set of qualitative data. A particular Link can connect assets with each other or to artifacts. Links can have different forms (cf. Figure 1) realized using typed properties (not shown in the meta-model): Functional links between production resources may represent a resource composition. Technical links may represent a wired connection from an Industrial PC (IPC) to a robot. To model a connection between an Asset and an Artifact, we use Trace Links, e.g., a robot controller requires a robot program. A Link can be manifested as Dependency, if the link is strictly required by an Asset. Assets, Artifacts, and Links can have attributes that describe characteristics of the particular object. These properties follow the 14.0 Asset Administration Shell (AAS) [3] design to facilitate the standardized representation of property views coming from several engineering disciplines. A Boundary object represents a pattern boundary that contains Assets and Artifacts, e.g., boundary (A).

These concepts provide the foundation to build an I4AN that explicitly represents PSE information and knowledge for a wide range of applications, such as change impact analysis. Figure 1 illustrates an I4AN with the relevant engineering artifacts and the links between the assets. This model can be created automatically by exploiting appropriate engineering data logistic systems [12]. This overall model can be the starting point to identify common reusable patterns [19].

**B. Patterns for I40 Asset Reuse**

This section describes four basic patterns for identifying best-practice candidates for reuse in their context. These identification patterns can be used as a starting point to identify patterns in the particular PSE contexts of domain experts.

The reuse of assets requires considering the asset itself and, beyond that, its embedding in the surrounding system and functional intentions [6], [9], [28]. As described, PSE comprises two main phases, rough and detail planning.

The rough planning phase consists of matching process skills required by products and provided by resources. This comparison shall be based on product creation (P1) and process execution (P2) patterns.

**P1. Product-Process-Skill Pattern.** Product creation in PSE aims at providing the combination of products with their requirements and processes to manufacture them. **Aim:** The Product-Process-Skill pattern (cf. Figure 1, tag A) supports product engineers in selecting appropriate processes for their products. This product creation pattern contains production processes with their input and output materials, boundary conditions, and required skills. **Solution:** The pattern can be identified by collecting all assets connected to the related processes by product-process-related links: For an output product isolate the input products and determine their relevant properties. For each input product determine the required process steps and build the aggregated required skills of the steps according to [10]. Group the products, process steps, and skills into a boundary object. For the outgoing and incoming links, determine whether they are strict dependencies. For dependencies, decide if you need to either expand the boundary...
or create a depending pattern object. **Example:** An example are screw-screwing combinations. We identified different reuse patterns from equivalence classes based on the screwing bit, the applicable torque, and the screw material (magnetic vs. non-magnetic) with industry partners.

**P2. Skill-Resource Pattern.** Process execution in PSE is to identify resources able to execute a production process based on their functional skills. **Aim:** The Skill-Resource pattern (cf. Figure 1, tag B) helps to select appropriate resources matching to the Product-Process-Skill pattern. This process execution pattern contains resources with their properties, boundary conditions, and provided skills. **Solution:** The pattern can be identified by collecting all assets connected to the related resource links: For a set of connected resources, determine their provided skills and properties. From the skills, build the aggregated provided resources skills according to [10]. Group the resources and skills into a boundary object and determine the dependencies. For dependencies, decide if you need to either expand the boundary or create a depending pattern. **Example:** The pattern supports the definition of skills, e.g., positioning, with predefined attributes, like positioning accuracy, which are fulfilled by a set of resources.

The main concern within the detail planning phase is realizing production resources providing all necessary functionalities to fulfill the required skills. Here, patterns related to resource structuring and functionality are relevant. Thus links shall be considered depending on the use case.

**P3. Resource-Resource Composition Pattern.** The goal of detailed engineering in PSE is detailing and programming the selected resources. **Aim:** The Resource-Resource composition pattern (cf. Figure 1, tag C) represents the composition of a resource from sub-components, with the knowledge on technical parameters and dependencies on the type and instance levels. A quality ensured resource tree pattern could be applied at this point, reflecting the optimized orchestration of resources. **Solution:** For a group of connected resources (part-of relation) determine which resources are required to either fulfill a particular skill or if they require each other for functionality. Group the strictly required resources into a boundary. For dependencies, decide if you need to either expand the boundary or create a depending pattern object. **Example:** Screwdrivers can be driven, e.g., electrically or pneumatically. Depending on the drive, the screwdriver requires a transformer for the current or not, which can be expressed in an RR pattern.

**P4. Resource-Artifact Pattern.** Within the commissioning phase of PSE the detailed resource system is established according to the relevant engineering artifacts, e.g., relevant for operation. **Aim:** The Resource-Artifact pattern (cf. Figure 1, tag D) aims at binding the required engineering artifacts to the resources used in the production system. This helps engineers to reuse resources and their corresponding data or programs as a bundle. **Solution:** From a resource, follow the trace links to the engineering artifacts. For the resource and the necessary engineering artifacts, use a boundary object to group them. For incoming or outgoing dependencies from resources or engineering artifacts, decide whether to expand the boundary or create a depending pattern object. **Example:** Screwdrivers have a minimum, maximum, and yield torque for a screwing process. The screwdrivers and function blocks controlling the torque of the screwdrivers can be expressed as a pattern and reused in future projects.

The use case Car Body with screwed-on parts can benefit from reuse patterns in (at least) four ways: (i) The product-process-skill pattern can support product engineers in selecting appropriate screwing processes for their car body parts (see tag A in Figure 1). (ii) The skill-resource pattern facilitates selecting appropriate screwdrivers to screwing processes (see tag B in Figure 1). (iii) The resource-resource composition pattern can be applied for the optimized combination of screwing resources, e.g., robots and robot controllers (see tag C in Figure 1). (iv) The resource-artifact pattern can be applied for reusing engineering artifacts, e.g., robot controllers and robot control programs (see tag D in Fig. 1).

**VI. FEASIBILITY STUDY AND DISCUSSION**

This section presents a preliminary feasibility study and discusses the contributions with a focus on the research questions raised in Section III.

**A. Preliminary Feasibility Study**

As a proof of concept, we used a part of the production system for the investigated use case “Car Body with Screwed-on Parts” from the initial domain analysis to design and instantiate the Industry 4.0 Asset Network (I4AN) in a Neo4J graph database. The I4AN was found easy to extract from existing engineering information, which has to be integrated according to the the I4.0 AAS design [3].

The graph database facilitated the effective and efficient exploration, querying, and visualization of the linked assets. In addition to the technical links between assets coming from engineering models, we instantiated dependency links between the assets. Deep domain expert knowledge has to be added to the I4AN manually. The concepts in the I4AN facilitated adding previously implicit domain knowledge to the graph. The I4AN instance associated to Figure 1 enables identifying 4.0 Assets that belong to a pattern for I4.0 Asset reuse (cf. Section VI-B). To investigate the functionality, we issued queries onto the I4AN to track the dependencies. We used iterative queries, similar to cause-effect graph exploration [33], starting at a selected I4.0 Asset, such as a skill, and followed the multi-model links to neighboring assets of a specified type until reaching a stopping condition. We were able to efficiently isolate parts of the I4AN that correspond to the basic patterns introduced in Section VI-B. This approach also worked for the reuse scenario system boundary analysis that can be translated into the question: Which set of dependency links connects a selected set of assets to their immediate neighboring assets? This capability indicates that engineers can utilize the I4AN to identify the network to identify familiar patterns of assets as candidates for reuse.

*Graph database Neo4J: https://neo4j.com*
B. Discussion

We conducted a domain analysis with 80 types of robot cells and 27 robot types. Further, we elicited requirements from domain experts at five European PSE companies. The requirements showed that a key aspect is modeling the multi-disciplinary dependencies between assets and engineering artifacts that need to be considered to identify reusable asset patterns. It is also essential to thoroughly model the boundaries of the patterns to allow suitable reuse in practice among the involved engineering disciplines.

RQ1a and RQ1b concerned models and dependencies that facilitate the identification of assets suitable for reuse. To address RQ1a and RQ1b, Section V-A introduced the Industry 4.0 Asset Network (I4AN) that addresses requirements R1 to R3 identified in Section IV. The I4AN builds on I4.0 assets and uses their administration shell to integrate property views from several engineering disciplines. In comparison to patterns in software engineering, this multi-disciplinary aspect adds complexity to identifying patterns for reuse in PSE.

We go beyond the state of the art [4], [18] by modeling skills as I4.0 assets using their digital representation for linking multi-disciplinary assets and identifying boundaries for reusable assets. We build on and go beyond [25] by integrating multi-disciplinary multi-model links between I4.0 Assets.

RQ2 asked which basic patterns for reuse facilitate the identification of patterns for reuse. To address RQ2, Section identified four basic patterns addressing requirement R4 (cf. Section IV). These patterns specifically incorporate regularly occurring connected assets in PSE that can be reused for similar problems. Therefore, they provide guidance for reuse design and management with the I4AN. In this sense, the I4AN provides designers with the capability to describe partial solutions and integrate partial solutions into a complete solution from production processes to automation devices that automate the production process.

Limitations. The following limitations require further investigation. The research in this paper focused on the reuse of production processes and associated automation system elements in a typical use case of automotive manufacturing, the Car Body with Screwed-on Parts use case. As we assume the findings of this paper to be relevant in the broader scope of production processes and automation system elements, e.g., for discrete production and continuous production, the approach should be investigated in a broader range of application areas.

The domain analysis was conducted by one of the paper authors with consultation from domain experts and checked for plausibility by the author team. While the feasibility study focused on a I4AN for a robot cell of typical complexity, the authors of this paper, consulting with domain experts in car manufacturing, conducted the design of the I4AN including dependencies that are missing in traditional PSE design. This reflects the current practice of PSE engineering only partially and introduced bias to the study, requiring validation in a range of traditional and advanced PSE environments.

VII. CONCLUSION AND FUTURE WORK

The Industry 4.0 (I4.0) vision of production systems that are easy to adapt depends on advanced capabilities for reusing proven production processes, I4.0 assets and software-intensive components that automate these production processes. In Production Systems Engineering (PSE), the reuse of I4.0 assets requires understanding the dependencies of these assets in multi-disciplinary systems-of-systems engineering with heterogeneous models.

This paper investigated the information requirements for advanced multi-disciplinary reuse scenarios, such as process and resource identification and for system boundary analysis. To address the challenges of scattered and implicit domain expert knowledge that may lead to overlooking risky dependencies of reusable system elements, we introduced the Industry 4.0 Asset Network (I4AN). The I4AN builds on the I4.0 Asset Administration Shell [3] design to integrate system element properties and dependencies from several engineering disciplines, such as mechanical, electrical, and software interfaces and technical links.

Therefore, the I4AN enables designing a knowledge graph that represents for a reuse scenario important multi-disciplinary dependencies between system elements as neighborhoods of I4.0 Assets. Further, the I4AN concepts facilitate representing domain expert knowledge that was implicit, e.g., to recommend using a resource type with a process type.

We presented the use case “Car Body with Screwed-on Parts” to illustrate typical I4.0 Assets and links in production processes and robot cells widely used in car manufacturing. In the I4AN of the use case (cf. Figure 1), we identified four types of patterns for reuse.

In a feasibility study, we evaluated the I4AN with reuse scenarios by instantiating an I4AN knowledge graph formulating scenario concepts and questions as data in and queries to the knowledge graph. The study results indicate that the I4AN model is a good foundation for PSE domain experts to identify patterns for reuse in their contexts.

The research results advance the state of the art in knowledge engineering in PSE by modeling the Skill concept as an I40 Asset. The I4AN provides a lens for analyzing similarities and differences in production process and system designs. To this end, we are providing the foundations for advanced reuse design and management with the I4AN and patterns.

The research results advance the state of the art by adapting blueprints for design pattern to a multi-disciplinary engineering environment where multi-model links are crucial. The I4AN provides designers with the capability to describe partial solutions and integrate these partial solutions into a complete solution, from production process to automation devices that automate the production process. The I4AN facilitates identifying risky external systems dependencies across several engineering disciplines as input to assess the reuse effort and risk of candidate solution designs.

Future Work. Validation of patterns for reuse. We plan to investigate I4AN applications for reuse to improve PSE tools, e.g., with knowledge on multi-model dependencies.
Scalability. We see the need to investigate the scalability of the I4AN in a larger context and with additional engineering disciplines to evaluate the impact on the multi-disciplinary dependencies and boundaries beyond the scale of work cells.

Skills. We consider examining the extended use of skills as an advanced method to abstract from process requirements to resource capabilities and their role in reusable process and resource assets, e.g., using standardized catalog search.

Extension of the I4AN with Semantic Web content. For the PSE domain, the I4AN seems well represented in a graph database as this technology is increasingly well accepted in PSE, while Semantic Web technology is mainly used in research. We envision extending the I4AN with knowledge organized with Semantic Web technologies, e.g., issues, recommendations as natural text. The I4AN knowledge graph can collect knowledge instances that can be converted efficiently to Semantic Web technologies to facilitate research on industrial data for Semantic Web researchers.

Security. Aggregating domain knowledge in an I4AN creates a high-value knowledge graph. This graph requires research on security concerns, e.g., theft of intellectual property or using it to plan attacks on systems that represent critical infrastructure.

ACKNOWLEDGMENT

The financial support by Christian Doppler Research Association, Austrian Federal Ministry for Digital and Economic Affairs, and National Foundation for Research, Technology and Development is gratefully acknowledged. The competence center CDP is funded within the framework of COMET – Competence Centers for Excellent Technologies by BMVIT, BMDW, and the federal state of Vienna, managed by the FFG.

REFERENCES


Software Design Pattern Analysis for Micro-services Architecture using Queuing Networks

Hanzhong Zheng, Justin Kramer, Shi-Kuo Chang
Department of Computer Science, University of Pittsburgh, Pittsburgh, PA, USA
{victorzhrz@cs.pitt.edu, jpk91@pitt.edu, schang@pitt.edu}

Abstract—Software design patterns are used to identify simple ways of realizing relationships among software entities or components for solving a commonly occurring problem. Design patterns allow the final software system to support different realized, non-functional requirements. In this paper, we are interested in three popular design patterns in Micro-services architecture: Fan (distributed), Chain, and Balanced, and study the influence of different system parameters to system performance. The simulation mimics system behaviors under specified design requirements for assisting software developers to select appropriate design pattern in software development life cycle (SDLC). In order to enable multi-pattern code generation, we extended our previous research on an automated modularity enforcement framework [1] from design pattern analysis to pattern evaluation.

Keywords – Micro-services, Design Pattern Analysis, Software Architecture Evaluation, Queuing Network Modelling

I. INTRODUCTION

Architectural patterns and design patterns are usually employed in the software development life cycle (SDLC). Software design patterns are “general and reusable solutions to a commonly occurring problem in software design within the context of software system design”. It helps the developers to communicate software architectural knowledge, bypass traps and pitfalls during the development process [2]. Usually, design pattern can only provide the templates or descriptions to the developers about how to solve problems in the process of designing an application or system, but cannot be directly transformed into code. To ensure the continuous delivery of trustworthy and high-quality software systems while reducing the burdens on programmers, design patterns become critical in the software development process. The current approach on employing design patterns has been focusing on object-oriented software design with emphasis on the relationship and interactions between classes or objects [2] [3] [4]. There have been a lack of emphasis on the design pattern for service-oriented architecture, especially for Micro-services. Micro-services consider an application to be a collection of loosely coupled, interconnected modular services, where individual services communication through REST APIs, and lightweight messages.

In this paper, we put focus on three popular design patterns in micro-service architecture. A software system is usually divided into several modules during the design phrase. To explicitly enforce the modularity in design patterns becomes very important in software system design. We extend previous work (automated modularity enforcement framework) for those design patterns in Micro-service architecture. The formal definitions of three top design patterns are [6]:

**Chain**: A “pipeline” layout of all components in the execution process. Clients establish one-to-one relationship with servers.

**Fan** (Distributed): All clients establish many-to-one relationship with a central database server. All the information will be stored into the central server.

**Balanced**: also known as Shared Data Pattern. Each client establishes one-to-one relationship with its database server. Multiple servers share their data.

We have conducted a comprehensive simulations using Queueing Network Modelling based tool, named Java Modelling Tool (JMT). The simulation results indicate that each design pattern has its own advantages for building appropriate software system products for satisfying proposed design requirements. The contributions of this paper are as followings:

1. We developed the automated design pattern analysis in Micro-service architecture from system pattern design to pattern evaluation.
2. We build a mathematical estimation model through parameterizing the expected cost for software system development
3. We conducted comprehensive simulation experiments and analysis for better understanding of each design pattern’s properties.

DOI Reference Number: 10.18293/SEKE2021-180
II. RELATED WORK

Micro-services Architecture. Micro-service architecture is a type of service-oriented architectural type, in which an application is constructed as a set of loosely coupled small services. The current practice of micro-service architecture mostly concentrates on the business enterprises such as Netflix, Amazon, e-commerce, etc. The next generation of micro-services requires ad-hoc tools for the creation of design patterns [5]. Sahiti Kappagantula introduces the several design patterns in Microservices architecture for providing reusable solutions to overcome common problems and improve application performance [6]. However, it is lack of the instructive and qualitative analysis among different patterns. In our work, we aimed to provide comprehensive analysis for different design patterns based on the performance quality attributes under a specific application scenario.

Queueing Network Modelling. Software architecture evaluation ensures an appropriate architecture is chosen for building complex software-intensive systems to any organization. Hence, architecture evaluation helps developers to ensure all stakeholders’ requirements have been satisfied. Commonly, software architecture evaluation can be classified into experience-based, simulation-based, mathematical modeling, and scenario-based [7]. The simulation-based evaluation approach usually is combined with mathematical modeling for estimating a more accurate system performance. However, it is not easy for evaluating system performance during the design process. Queueing Network Modelling represents the computer system as a network of queues, and analytically evaluates the system performance [8]. It can simulate a group of service centers, which can make use of our three different patterns. Macro Bertoli et al. presented Java Modelling Tools (JMT) suite for evaluating system performance using queueing models [8]. JTM integrates a graphic user interface and other methodologies such as discrete event simulation, bottleneck identification in multiclass environment, etc. We simulated the performance results of our three different design patterns using JMT with the same initial parameters setting.

III. MATHEMATICAL MODELLING OPTIMIZATION

The software cost estimation not only can minimize the total cost of software development cost, but it also ensures the final product can satisfy the requirements, which generally refer to the quality attributes such as performance, functionalities, etc. Many estimation models have been developed and widely used. In general, there are two major categories of existing models: algorithmic and non-algorithmic.

Algorithmic cost modelling uses mathematical expressions to predict the development costs based on the estimations of system size, complexity, and other process and product factors. Finding the most appropriate expression can estimate software development costs, which are important for analyzing the performance of our three different structural patterns with keeping a relatively low development cost. The general form of an algorithmic cost estimation can be expressed as:

\[ E[\text{effort}] = A \times S^B \times M \]  

where \( A \) is constant factor that depends on the type of final software product, \( S \) is the code size of the software or functionalities of certain components, \( B \) is the exponential factor that usually lies in range of \([1, 1.5]\), indicating the fact that costs do not linearly increase with project size, and \( M \) is a constant multiplier for combing process such as dependability requirements. Our automatic code generation tool is built based on Micro-service applications. The service reliability is critical for service-oriented system. We added a reliability modeling term to the cost estimation expression as following:

\[ E[\text{effort}] = A \times S^B \times M + \sum_{i=1}^{n} (1 - \frac{f_i}{t_i}) \times c_i + \epsilon \]  

where the term \( 1 - \frac{f_i}{t_i} \) is the simplified probability model of estimating service reliability from, \( f_i \) is the number of services executions without exceptions occurrence; \( t_i \) is the total number of service invocations. \( c_i \) is the anticipated cost of executing service. During the simulation, we assume that the probability of software system failure as a stochastic process. In most logarithmic cost models, the code size (\( S \)) is usually difficult to estimate when the specifications are not available. Since factors \( B \) and \( M \) are usually subjective, we are mainly interested in their relations to the cost estimate model during the optimization process. To study the relation between \( M \) and \( E[\text{effort}] \) we take the partial derivative with respect to \( M \):

\[ \frac{\partial E[\text{effort}]}{\partial M} = A \times S^B \]  

Similarly, we take the partial derivative with respect to \( B \):

\[ \frac{\partial E[\text{effort}]}{\partial B} = A \times M \times S^B \times ln(S) \]
Therefore, in the simulation experiment section, we study the changing of $M$ or $B$ variables, while keeping other variables to be constant to determine their effects to the cost and system performance.

IV. EXPERIMENT TOOL

we extend our previous work by adding extra functionalities and input parameters to support the three patterns. This allows software developers select their preferred architecture after testing the performance of each design pattern. Our key objective was to add a layer of experimentation and design to our automatic code generation. We added a cleaning functionality to AutoGenerator, which allows users to experiment with combinations of design patterns and service creation without the risk of damaging their templated product. Therefore, our ‘cleaning’ function provides users with flexibility in their designs and improve software development process. After we established our ‘cleaning’ functionality, we started on the creation of each micro-service architecture design pattern. The first stage of production was analysis of each design pattern. The results of our design pattern analysis led to formulated architecture diagrams. In Fig. 1, we demonstrate the balanced design pattern for our micro-service architecture in medical application. In the architecture diagram, a balanced design pattern is followed by combining the fan and chain design patterns. In the figure 1, the doctor and patient services act as examples of the fan design pattern. The doctor and patient services connect to our centralized registration service through HTTP communication via their localized micro-service servers. Within the fan design pattern, each micro-service will contain its own database, server to communicate with the central registration server, and controller layer. Also, the prescription service acts as an example of the chain design pattern. In the chain design pattern, our AutoGenerator establishes the formation of a new micro-service consisting of a service layer and a database. The chain design pattern demonstrates the pipeline execution pattern, so prescription service acts as an addition to the functionality of the fan structure’s patient service. In this example, the patient service will query the prescription service directly through a public interface to acquire information about the prescriptions of a specific patient. IServices are assigned a unique probability to appear as micro-services within the fan or chain design patterns. The element of probability in the balanced design pattern allows for permutations of samplings to appear as the result of the auto generation. Thus, basic analysis of each design pattern allowed for translation into a micro-service architecture for our AutoGenerator.

Fig. 1: A balanced micro-service architecture utilizing both chain and fan pattern design.

Our process begins with IC cards, which define the service interactions witnessed in the architecture being designed [1]. Once the IC cards are defined, the ICMS can output an XML specification as shown in Fig. 2. The XML specification specifies the structure of different software components and initial system parameters, which all are used in the system simulations.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<icCardList
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <icCardEntry icEntryId="2788" icEntryName="ex3">
```

47
Based upon the XML specification of the IC cards, the AutoGenerator can then create the output modules. Fig. 3 demonstrates a portion of the code to generate these modules, which allows for several key variations. The first variation is the parameter that designates the design pattern to utilize. The input parameter may be selected as “balanced”, “fan”, or “chain” depending on the use case. Furthermore, each design pattern follows a set of generation rules which establishes modularity and provides a basis to fill in implementation details for the architecture. For the chain pattern, our AutoGenerator builds a simplified micro-service layer composed of the necessary modules to connect a Micro-service to its database and to another Micro-service. The necessary modules are comprised of a data repository interface and a method-layer interface to interact with the service. For our balanced pattern, we follow the directed creation of a fan or chain implementation of the auto generation based upon probability. The same skeleton of our AutoGenerator now contains flexibility in its design pattern and the ability to enable hypothesis testing through cleaning function.

```python
for i, service in enumerate(services):
    if tables[i]:
        separated_table = tables[i].split('(
        adjusted_columns =
            adjust_columns(separated_table[1])
        TABLE_NAME = separated_table[0]
```

V. QUEUING NETWORK MODELLING ANALYSIS

We decide to use Queuing Network simulation for evaluating the different patterns’ performance. The layout designs of the patterns are illustrated in Fig. 4 using the JMT simulation software. All the simulation layouts reflect the definition of different design patterns in the beginning of the paper. JMT contributes to perform system evaluation studies in the following two ways: 1. Statistically analysis such as confidence interval analysis, variance estimation, etc. 2. A friendly user interface for the description of system and parameters analysis. The main parameters related to the interested variables: system size ($B$) and multiplier of system combining process ($M$). Since we assume that the micro-service design patterns are targeted to medical application, so that $M$ can be reflected the requirements of patients’ service demand time, and $B$ is the doctor service time. The $S$ is the application that handles the number of patients in the system evaluation (Arrival rates): Exponential, Normal, Uniform, etc. In this paper, we use the normal distribution for better modelling the number of patients in real case. Service Demand: the average amount of time (workload) that each user/patient required for the doctor’s service (on user/patient side). Queueing disciplines: 1. Non-preemptive: First Come Frist Served (FCFS), Last Come
First Served (LCFS), Random (RAND), etc. 2. Preemptive: Server sharing, Discriminatory Server sharing, etc. Routing of the users/patients in the system: the current setting is the Round Robin, which simulates that there is a waiting room for the patient to visit doctor in our system. Service Time: the maximum amount of time that doctor to diagnose each patient (on doctor side). User Preference: the developers’ inclining towards to specific structure patterns. This allows the user to select its own desired patterns.

We converted the design of patterns into well-structured XML code to input of simulation with specified components layout and system parameters. The simulation settings are: (1) The number of users is continuously increasing; (2) All the servers have the limited amount of disk capacity; (3) Each user submits the jobs according to a normal distribution with parameters different parameters; (4) The evaluation metrics are Throughput (# of jobs/second), Queuing Time (sec/user), Response Time (second/job), System utilization (# of working jobs/second); (5) Each simulation lasts until the model converges and we conducted 15 repeat runs making sure the accurate final performance results of each pattern. The experiment results are fall into with relative error < 0.03. Our purpose of the simulation is to find an appropriate structural design pattern, which plays a critical role in software development process. We alternate different parameters for different patterns during the simulation. Figures 5, 6, and 7 show the performance results of the 3 different patterns under the different simulation inputs. The alternations of the desired variables enable to reflect the behaviors of different patterns. The simulation can see the fluctuations of different patterns under the different parameters’ setting. This allows the developers to decide appropriate design pattern during the development process.

Fig. 4 The top, middle, and button figures are the simulation layouts for distributed, balance, and chain pattern.

Figure 5: the performance of three patterns with the increase of service demand from 30, 50, 100.

Figure 6: the performance of three patterns with the increase of service time from 50, 100, 200.
We treat each user is managed by a process in the system, which is related to the system requirements in operating and combing user processes. The simulation results show that the throughput of all patterns decreases with the increase of all the parameters. The FAN pattern largely impacted by the service time and the Chain pattern largely influenced by the number of users in the system. The FAN pattern has a higher and higher value in the System utilization, which represents that there are increasing number of working jobs in the system. The increase of the service time surprisingly decreases the system utilization of Balanced pattern, while the rest of two patterns both increase. Our explanation is the routing problem due to the number of data servers and components. Since there are multiple servers in the Balanced pattern, it brings the higher capacity to handle the dramatically increase of users in the system. However, the data replication and synchronization become the main challenge in this design pattern.

As for the FAN (distributed) pattern, it maintains a comparatively reasonable performance under different parameters setting, but it can easily be influenced by the alternations of parameters. The centralized data server avoids problems in other two patterns but requires a more intelligent routing and queuing discipline for handing users in “burst” situation. The choice of different design patterns depends on the design requirements and also takes the user’s preference into the consideration. The Chain pattern involves the structure of “pipeline” design. The execution of each process is strictly followed the order, which unavoidably cause the “stalls” inside the execution pipeline. However, certain applications such as online patient diagnosis in medical domain has the preferences on chain pattern.

VI. CONCLUSION AND FUTURE WORK

This paper proposes a new software design approach using the Micro-service architecture. The extended automated code generation framework enables code generation under three different design patterns. We compared our design patterns using Queuing Network modelling for performance analysis. The queuing network allows for analytic study on the software system, which is represented as a network of queues with collections of service centers. We compared the performances of the different design patterns under different parameters settings and provided an analytical evaluation for them. Our next goal is to study the influence of parameters (such as the capacity of the servers, data usage volume, user preferences of specific patterns and so on) to the performance of different design patterns.

REFERENCES


Correctness Arguments for an SDN MAC Learning Algorithm

Liang Hao, Xin Sun, Lan Lin
Department of Computer Science
Ball State University
Muncie, IN 47306, USA
{lhao, xsun6, llin4}@bsu.edu

Abstract

The emerging software-defined networking (SDN) paradigm is a radical departure from the traditional networking architecture as it decouples the control plane and the data plane and also centralizes the control plane. While SDN has proven to provide an unprecedented opportunity for creating advanced control functions that are capable of optimizing global properties of a network, recent works including our own have shown that the implementation of such control functions is a software engineering challenge on its own due to the unique architecture of SDN. In our recent work [7], we have taken the first step towards tackling this challenge by presenting a preliminary case study of implementing the classic Media Access Control (MAC) learning algorithm for SDN. A major discovery we made is that, for a proper implementation of the algorithm, the algorithm itself must be modified to account for the architectural differences of SDN. This paper builds upon and advances our prior work, by presenting a detailed description and analysis of the SDN version of the MAC learning algorithm, to justify the modifications as necessary for a correct and efficient implementation. As such this paper sheds light on the interplay between the algorithm design and software implementation of control functions for SDN, which we believe is an important contribution to both the software engineering community and the networking community.

1 Introduction

Software-defined networking (SDN) is a new paradigm in the networking landscape that transforms the way networks are constructed and managed. A defining architectural feature of SDN is the centralized control plane, which is a radical departure from the traditional networking architecture where the control plane is fully decentralized. The control plane serves as the brain of a network: it controls how packets are forwarded by producing the necessary forwarding tables. The data plane, in contrast, handles the packet streaming by looking up those tables.

In the traditional networking architecture, both the control and the data planes are implemented on every networking device (e.g., routers and switches), and hence they are fully decentralized and tightly coupled. In contrast, the SDN control plane is implemented by a logically centralized software platform, called the network OS or the controller, with all the application programs (called SDN apps) running on top of it. The controller, together with all the SDN apps, runs on a PC server that connects to the networking devices (called SDN switches). The SDN data plane is implemented by the packet forwarding hardware on all the SDN switches. A communication protocol, called OpenFlow [16], is used for the controller to communicate with the SDN switches and vice versa. The SDN architecture is illustrated in Figure 1.

The unique architecture of SDN provides an unprecedented opportunity for creating advanced control functions, but at the same time poses a significant challenge to the implementation of such functions as SDN apps. On the one hand, the centralized control plane oversees the entire network and offers a global view for creating control functions
capable of global optimization. On the other hand, the decoupling results in the control plane’s limited visibility into the data plane. In particular, only a small subset of packets being streamed in the data plane are copied to the controller, so as not to overload the controller or the network. This poses challenges to the implementation of SDN apps.

SDN research has largely focused on the algorithm design of control functions. However, relatively little attention has been given to the software engineering aspect, which we believe is an important research problem worth addressing on its own. As a first step, in our recent paper [7] we presented a case study of implementing the classic Media Access Control (MAC) learning algorithm (an essential link-layer forwarding algorithm implemented by every traditional Ethernet switch) in the SDN architecture. A major discovery we made in [7] is that an efficient implementation requires modifications to the algorithm itself due to the architectural differences of SDN. We presented in [7] such an implementation following rigorous software specification and design methodologies. However, a detailed description of the modified algorithm itself or an analysis of the algorithm to justify the modifications was not presented in [7], due to the fact that the focus of that paper was on applying rigorous software specification and modular design to derive an implementation through stepwise refinement.

Building upon and advancing our prior work, the primary contributions of this paper include a detailed description of the SDN version of the MAC learning algorithm, and a detailed analysis that shows why the modifications are necessary for the SDN architecture. We hope this paper will shed light on the interplay between the algorithm design and the software implementation of SDN control functions.

The rest of the paper is organized as follows. Section 2 presents a description of the classic MAC learning algorithm. Section 3 explains why a direct and straightforward migration of the classic algorithm to SDN is impractical, presents the SDN version of the algorithm, and argues why the modifications are necessary for achieving a correct and efficient implementation. Section 4 discusses related work and Section 5 offers concluding remarks.

2 The Classic MAC Learning Algorithm in Traditional Networks

MAC learning is the classic link-layer algorithm that enables an Ethernet switch to perform two essential functions: (i) constructing and updating the switch table (a control plane function), and (ii) using the table to forward packets toward their destinations (a data plane function) [8]. In the traditional networking architecture, this algorithm runs on every Ethernet switch.

The switch table produced by the algorithm is stored in the switch memory. An entry in this table contains the MAC address of some device in the network, the switch port believed to lead toward that device, and a timer. The MAC address is used as the key for indexing the table (MAC addresses are all unique), and the timer is used to remove the entry in the future if it becomes stale.

On each Ethernet switch, whenever a packet is received on a port, the algorithm performs the following two steps (illustrated in Figure 2):

Step 1: the algorithm extracts the source MAC address s from the packet header. If s is not in the switch table, it creates a new table entry \((s, p, t)\) where \(p\) is the incoming port (which must lead toward \(s\)) and \(t\) is a timer set to expire after some period of time. If such an entry already exists, the timer will be reset. If there exists an entry with the same MAC address \(s\) but with a different port, the port will be updated to \(p\) and the timer be reset.

Step 2: the algorithm uses the destination MAC address \(d\) in the packet header to look up the switch table. If an entry with \(d\) exists, the packet will be sent out the associated port. If such an entry does not exist, the algorithm will flood the packet out all active ports except the incoming port \(p\).

In addition, the algorithm will delete a table entry when the associated timer expires. This is to get rid of any potentially stale entry due to topology changes, e.g., hosts being removed or relocated.

We note that the proper functioning of Step 1, specifically the resetting of timers, relies on the fact that the algorithm sees every incoming packet. As we will explain next, this is not the case in the SDN architecture.

3 The SDN Version of the MAC Learning Algorithm and Correctness Arguments

We first describe the key difference of the SDN architecture and explain why a direct and straightforward migration of the MAC learning algorithm to SDN will not work. We
then describe several modifications to the algorithm and argue why those modifications are necessary for achieving a correct and efficient implementation. Lastly, we use an example to illustrate the operation of the new algorithm.

3.1 A Direct Migration Is Inefficient

The control plane and the data plane are completely decoupled in the SDN architecture. The control plane is implemented by the software controller running on a centralized server; and the data plane is implemented by all the decentralized SDN switches. Implementation of the MAC learning algorithm will involve the controller and all the SDN switches, as Step 1 of the algorithm is a control plane function and Step 2 a data plane function. The two parts of the algorithm will need to interact via the OpenFlow communication protocol [16]. A direct migration of the classic MAC learning algorithm to SDN would work as follows.

On the controller: Step 1 of the algorithm would be implemented as an SDN app running on the controller. The app constructs and updates the switch table for every switch. For each switch $s$, the app collects every incoming packet together with the incoming port information. For each packet the app either creates a new switch table entry or updates an existing entry. It then sends the new or updated table entry to $s$ in an OpenFlow protocol message. (The OpenFlow protocol messages are called “FlowMod.”)

On every SDN switch: Whenever a switch receives a new packet, it sends the packet together with the incoming port information to the controller. The switch then receives a new or updated table entry from the controller and caches it locally. The switch uses the destination MAC address in the packet header to look up its locally-cached table, and either forwards the packet to the associated port or entry is found, or floods the packet on a table miss.

In addition, the SDN app is responsible for creating and resetting the timers. When a timer expires, the SDN app deletes the associated table entry and informs the switch to delete the same entry from its locally-cached table.

This direct and straightforward migration of the MAC learning algorithm to SDN unfortunately will not work for two reasons. First, requiring every switch to send every packet to the controller creates substantial traffic overhead; for every packet the controller will send a FlowMod message back to the switch, further increasing the overhead. Second, the controller will receive and process every packet; the amount of link bandwidth and CPU power required to implement such a controller is simply beyond practicality.

3.2 Necessary Algorithm Modifications

To address the above-mentioned efficiency issues, our prior work [7] made several important modifications to the MAC learning algorithm when implementing it for SDN.

First, the modified algorithm requires that, whenever an SDN switch receives a packet, it should first look up its locally-cached table using the source and destination MAC addresses in the packet header and information regarding the incoming port. Note that each entry of the locally-cached table maps a three-tuple (source address, incoming port, destination address) to a destination port. If a table entry is found to match the three-tuple (source address, incoming port, destination address), the switch should forward the packet out the associated port without sending the packet to the controller. Only upon a table miss should the switch send the packet and information regarding the incoming port to the controller.

Second, the MAC learning app running on top of the controller will construct and update the forwarding tables for all the switches. Whenever the app receives a packet and incoming port information from a switch $s$, it either creates a new entry in the table of $s$ if the source MAC address of the packet does not exist in the table, or updates the existing entry based on the new port information. It then uses the destination MAC address of the packet to look up the table of $s$. If an entry exists, it will send the entry to $s$ in a FlowMod message; otherwise, it will instruct $s$ to flood the packet, also in a FlowMod message.

While these changes dramatically reduce the number of packets sent to the controller, they create two new problems for the implementation, which we discuss below.

Implementation of the timers associated with table entries: the controller no longer sees all the packets being forwarded on all the switches, and thus it is unable to properly reset the timers in the switch tables. This means that the timers cannot be maintained by the controller.

Synchronization of table entries between the controller and the switches: even after the controller deletes an entry $(h, p)$ in the forwarding table of switch $s$ as the entry may become stale (e.g., the host with MAC address $h$ may have relocated and no longer be reachable from switch $s$ through its port $p$), $s$ may still have a locally-cached table entry containing $(h, p)$ as the destination address and outgoing port; this will cause $s$ to continue to forward any packet destined to host $h$ out of port $p$, resulting in permanent loss of those packets.

To solve both problems, several additional modifications to the MAC learning algorithm are necessary. First, whenever the controller sends a FlowMod message to a switch that contains a table entry, it also creates and sends a reversed FlowMod message, in which it flips source address with destination address, and flips incoming port with outgoing port. For instance, if a FlowMod message (to be sent

1 The locally-cached switch table is also called the FlowMod table.
to switch $s$) with source address $sh$, incoming port $sp$, destination address $h$, and outgoing port $p$ is created, the controller also creates a reversed FlowMod message (to be also sent to switch $s$) with source address $h$, incoming port $p$, destination address $sh$, and outgoing port $sp$. By the time they are created, it must be the case that $(h, p)$ and $(sh, sp)$ are both valid table entries, i.e., neither $(h, p)$ nor $(sh, sp)$ is a stale host-port pair.

Second, switches maintain the timers for all the locally-cached table entries, as switches see all the packets and can thus properly reset the timers. The controller on the other hand does not maintain any timer.

Third, whenever a locally-cached FlowMod table entry, with $(sh, sp)$ as (source address, incoming port) and $(h, p)$ as (destination address, outgoing port), gets deleted due to timer expiring, the switch sends a FlowRemoved message notifying the controller. The controller then deletes $(sh, sp)$ from the lookup table, and sends a FlowMod message instructing the switch to delete the reversed entry with $(h, p)$ as (source address, incoming port) and $(sh, sp)$ as (destination address, outgoing port) from its locally-cached table. Therefore, in case any host-port pair $(h, p)$ goes stale, i.e., host $h$ is no longer reachable from port $p$, for any locally-cached table entry with $(h, p)$ as (destination address, outgoing port), its reversed table entry, in which $(h, p)$ is the (source address, incoming port) pair will eventually expire and get removed, leading to the former table entry with a stale (destination address, outgoing port) pair being removed as well.

### 3.3 Illustrating the Algorithm

We illustrate this subtlety of the SDN MAC learning algorithm using an example. Assume there are two hosts in the network with two SDN switches (Figure 3). Host $H_1$ is connected to switch $S_1$ through port $a$. Host $H_2$ is connected to switch $S_2$ through port $b$ on $S_1$ and port $c$ on $S_2$. The SDN controller maintains lookup tables for $S_1$ and $S_2$, which are initially both empty. The two switches each maintain their own FlowMod tables, which are also initially empty.

Assume $H_1$ first sends a packet to $H_2$ (Figure 4 sub-step (1)). $S_1$ consults the SDN controller for how to handle this packet (sub-step (2)). The controller adds $(H_1, a)$ to $S_1$’s lookup table (sub-step (2)). The controller instructs $S_1$ to flood the packet (sub-step (3)). The flooded packet arrives at $S_2$ though port $c$ (sub-step (4)). $S_2$ consults the controller resulting in $(H_1, c)$ being added to $S_2$’s lookup table (sub-step (5)). The controller instructs $S_2$ to flood the packet (sub-step (6)). The flooded packet gets to $H_2$ (sub-step (7)).

Now assume $H_2$ sends a packet back to $H_1$ (Figure 5 sub-step (1)) and assume the migrated algorithm didn’t enforce reversed FlowMod messages to handle the subtlety introduced by stale (host, port) entries. $(H_2, d)$ is added to $S_2$’s lookup table as $S_2$ consults the controller for how to handle this packet (sub-step (2)). Since the destination
address $H_1$ is already in $S_2$’s lookup table, the controller instructs $S_2$ to forward this packet to port $c$ (sub-step (3)). $S_2$ caches this instruction in its FlowMod table (sub-step (3)). The forwarded packet gets to $S_1$ through port $b$ (sub-step (4)). Now it’s $S_1$’s turn to consult the controller, which leads to another entry being added to $S_1$’s lookup table (sub-step (5)). The controller instructs forwarding, and the instruction is cached in $S_1$’s FlowMod table (sub-step (6)). Finally, the forwarded packet gets to $H_1$ (sub-step (7)).

Next we assume $H_1$ is disconnected from $S_1$ and connected to $S_2$ through port $f$, while $H_2$ continues sending more packets to $H_1$ (Figure 6). This topology change is not observed by either switch, as packets are being received through the same (source address, incoming port) pair, and forwarded to the same destination using the cached FlowMod table entries. Notice that all such packets from $H_2$ to $H_1$ will be lost with neither the switches nor the SDN controller being aware of the underlying topology change of the network and the resulting permanent packet loss.

With the correct algorithm that enforces reversed FlowMod messages, in Step 2 any forwarding instruction from the controller to the switch enforces a FlowMod table entry, together with its reversed FlowMod table entry, to be simultaneously cached on the switch (Figure 7 sub-steps (3) and (6) and shaded FlowMod table entries). In Step 3 after $H_1$ moves to the new location (Figure 8), although there will be a little packet loss at the beginning, soon this will be fixed. This is because in Step 4 (Figure 9) the FlowMod table entries on both switches for the reversed FlowMod messages (sent earlier by the controller) will expire (after $H_1$ changes the switch it is connected to). This results in two lookup table entries being deleted from the controller, and instructions from the controller to both switches to delete the remaining two FlowMod table entries. Finally in Step 5 (Figure 10) subsequent packets from $H_2$ to $H_1$ are consulted with the controller and flooded, reaching $H_1$ through port $f$. The first packet from $H_1$ through port $f$ will be seen by the controller, which will add an updated lookup table entry for both switches with $H_1$ as the source address. Future packets from $H_2$ to $H_1$ will be forwarded correctly by the controller using the updated lookup table entries.

### 4 Related Work

Since the inception of SDN around a decade ago [9], the research in SDN has primarily focused on (i) creating novel control functions, such as load balancing [6], power saving [1], anomaly detection [14], etc.; these works mostly focused on designing the algorithms and protocols, and not on the implementation; (ii) the orchestration of multiple control functions running on the same software-defined network [2, 3, 10, 15, 17], and (iii) improving the performance,
reliability, and security of both SDN switches [5, 11, 12] and controllers [4, 13]. However, relatively little attention has been given to the software engineering aspects of SDN apps. Our recent work [7] took the first step in tackling the software engineering challenges of implementing control functions as SDN apps, through implementing the classic MAC learning algorithm for the SDN architecture. In contrast, this paper presents a detailed analysis of the SDN version of the algorithm that looks deeper into the subtleties in algorithm design required for a correct and practical implementation. To the best of our knowledge, this is a first work that explicitly studies the interplay between algorithm design and software implementation of SDN apps.

5 Conclusion

This paper presents a detailed analysis of the MAC learning algorithm and its implementation in the context of SDN. We show that a direct and straightforward migration of the classic MAC learning algorithm to SDN will not work, due to its substantial bandwidth overhead and the impractical workload placed on the controller. We then present a modified version of the algorithm that takes into account the SDN architecture. Through a detailed analysis we show that the modifications are necessary for a correct, practical and efficient implementation without impacting network performance. We believe this work will be of interest to both the software engineering and the networking communities. We plan on applying the new understanding developed in this study to migrating more algorithms and protocols to the SDN environment.

Acknowledgments

This work was generously funded by Air Force Research Laboratory through the National Science Foundation (NSF) Security and Software Engineering Research Center (S^2ERC), and by the NSF under Grants CNS-1660569 and 1835602.

References


Multiclass Classification of Four Types of UML Diagrams from Images Using Deep Learning

Sergei Shcherban and Peng Liang*
School of Computer Science
Wuhan University, China
k1myriel@yandex.ru, liangp@whu.edu.cn

Zengyang Li
School of Computer Science
Central China Normal University, China
zengyangli@ccnu.edu.cn

Chen Yang
IBO Technology (Shenzhen) Co., Ltd.
Shenzhen, China
c.yang@ibotech.com.cn

Abstract—UML diagrams are a recognized standard modeling language for representing design of software systems. For academic research, large cases containing UML diagrams are needed. One of the challenges in collecting such datasets is automatically determining whether an image is a UML diagram or not and what type of UML diagram an image contains. In this study, we collected UML diagrams from open datasets and manually labeled them into four types of UML diagrams (i.e., class diagrams, activity diagrams, sequence diagrams, and use case diagrams) and non-UML images. We evaluated the performance of five popular neural network architectures using transfer learning on the dataset of 3231 images that contains 700 class diagrams, 454 activity diagrams, 651 use case diagrams, 706 sequence diagrams, and 720 non-UML images, respectively. We also proposed our neural network architecture for multiclass classification of UML diagrams. The experiment results show that our proposed neural network architecture achieved the best performance amongst the algorithms we evaluated with an accuracy of 98.65%, a precision of 96.76%, a recall of 96.48%, and an F1-score of 96.62%. Moreover, among the neural network architectures that we have evaluated, our proposed architecture has the least parameters (around 2.4 millions) and spends the least time per image (0.0135 seconds per image using GPU) for classifying UML diagrams.

Keywords—UML Diagrams, Neural Network, Deep Learning, Multiclass Classification

I. INTRODUCTION

To define and communicate the design and architecture of systems, software engineers are challenged by the increasing complexity of software systems, especially when groups of developers use different modeling notations in distributed development. Software developers, in particular architects and designers, are therefore using Unified Modeling Language (UML) diagrams as a universal set of notations to promote the concept description and collaboration in the development of software systems. UML was stemmed as the union of three object-oriented design practices: the Booch method [2], the object modelling technique [3], and the Objectory method [4]. The UML standard was published and maintained by the Object Management Group (OMG). In addition, both in industry and academia, the use of UML diagrams as a standard for developing software [5]. Class diagrams, activity diagrams, sequence diagrams, and use case diagrams are the most common four types of UML diagrams used in industry [6]. UML class diagrams demonstrate the classes in a system, attributes and operations of each class, and the relationship between classes, and act as a focal role in defining software structure. Activity diagrams explain how activities are organized to deliver a service that can be at different levels of abstraction. Sequence diagrams are diagrams of interaction that detail how operations are done, and they capture the interaction between objects. Use case diagrams provide description of the interactions between users and the system.

In the research concerning UML modeling, researchers need repositories with a large number of samples of UML diagrams [7] [8] [9], such as the Lindholmen dataset [10]. These datasets can also be used to create helpful tools for developers, such as creating UML diagrams by using natural language specifications [7] or using UML diagrams to produce code [8]. Images are one of the most used ways of storing and sharing UML diagrams. Therefore, identifying whether an image belongs to UML diagrams or not is the issue of constructing such repositories. Non-UML images are often found, especially in large datasets, such as the Lindholmen dataset [10]. Sorting through thousands of pictures manually requires a significant amount of time and effort. It is therefore necessary to identify various UML diagrams from images automatically. Recently, many researches have been focused on classifying images by applying deep learning techniques (e.g., [11] [12] [13]), and practitioners and researchers are making these techniques accessible for use. Besides, in terms of precision of image classification, deep learning algorithms outperform classical machine learning algorithms [14]. Also, modern deep learning frameworks (such as Tensorflow or PyTorch) include the ability to use the GPU for neural network training and inference, which accelerates the use of deep learning-based image classification approaches.

In this paper, we presented an approach to identify various types of UML diagrams automatically from images using deep learning algorithms. Initially, we gathered a dataset of 3231 images (700 class diagrams, 454 activity diagrams, 651 use case diagrams, 706 sequence diagrams, and 720 non-UML images). An experiment was then performed with several popular neural network architectures [15] [16] that can be found in current deep learning frameworks [17] [18] and are widely used for applying neural networks (i.e., MobileNet [19], DenseNet [20], NasNet [21], ResNet [22], Inception [23]). We...
evaluated these selected neural networks with semi-trainable transfer learning (the convolutional part of pre-trained neural networks was not trained) and fully-trainable transfer learning. Also, we proposed a new neural network architecture for multiclass classification UML diagrams.

The contribution of this paper is threefold: (1) a dataset which contains UML class diagrams, activity diagrams, use case diagrams, sequence diagrams, and non-UML images, serving as a starting point for researchers to further investigate the UML diagram identification and classification problem, (2) an approach to automatically identify various types of UML diagrams by using deep learning techniques, and (3) a neural network architecture which can fast and effectively classify various types of UML diagrams from images.

The rest of the paper is organized as follows: Section II introduced related works. Section III describes the research questions, the data collection, the classification process, and the evaluation process, and the experimental setup. Section IV provides the results. Section V presented the threats to validity. Section VI describe conclusion of this work with further work directions.

II. RELATED WORK

Recently, various techniques to extract features from diagrams have been introduced. Karasneh and Chaudron [24] introduced a process to extract UML class diagrams from images and transform them into XMI format. Fu and Kara [25] proposed a method for converting engineering diagrams into connected graphs. Ho-Quang et al. proposed an approach to classifying UML class diagrams from images automatically by using machine learning algorithms and different feature extraction techniques [9]. Despite its effectiveness, this method to image classification consumes 5.84 seconds per image, which can be problematic when using it with big datasets. Mohd Hafeez Osman et al. showed that reverse-engineered and forward-engineered UML class diagrams can be classified by using machine learning [26]. Ahmed and Huang also applied machine learning to classify role stereotypes of UML class diagrams in order to quickly get the knowledge about role stereotypes for developers [27]. They achieved an accuracy of 89.6% in the multiclass classification of role stereotypes of UML class diagrams using Random Forest with SMOTE oversampling. Rashid classified UML sequence diagrams by applying machine learning and computer vision algorithms to facilitate the creation of repositories containing UML diagrams [28]. His work achieved an accuracy of 90.8% using the methods from the OpenCV framework, such as Canny edge, probabilistic Hough lines transform, and FindCountors as feature extraction methods. Bian et al. proposed an approach to automatically grade students’ UML class diagrams by using semantic, structural, and syntactic matches between the teacher’s solutions and the students’ solutions [29]. They received a variation of 14% between the teacher’s grade and the grade received by using their tool by grade 20 students.

Because neural networks do not demand additional feature extraction algorithms for classification and preform feature extraction automatically using convolution layers [11], their classification speed is higher than that of approaches that used combination of feature extraction algorithms and classical machine learning algorithms. Moreover, in some computer vision tasks, such as object recognition, modern neural networks outperform humans in terms of accuracy [12].

In deep learning, the idea of reusing the acquired knowledge from one task to another is called transfer learning [13]. It has been proven as an effective way to implement neural networks without using a lot of resources on training and searching for a neural network’s architecture. Transfer learning means applying the neural network’s weights obtained within one task to complete or partially complete training on a new task.

To the best of our knowledge, there are no works that are directly aimed at the automatic multiclass classification of UML diagrams. Thus we open this line of research by proposing an approach to UML diagram classification and making available to the public the dataset used for the experiments.

III. RESEARCH DESIGN

This research aims to study how we can automatically classify multiclass UML diagrams by using deep learning algorithms. In this study, we investigated four Research Questions (RQs):

**RQ1: What is the best performance of semi-trainable transfer learning for multiclass classification of UML diagrams?**

**Rationale:** This RQ aims to get the best classification algorithm (in terms of performance) when training algorithms to recognize UML diagrams. Various deep learning algorithms can produce different results, depending on the architecture, configurations, and datasets used. Our selection fell on five deep learning architectures, including MobileNet, DenseNet169, NasNetMobile, ResNet152V2, and InceptionV3, because they are commonly applied in image classification and can be founded in modern deep learning frameworks. We evaluate the performance of semi-trainable transfer learning of each algorithm using accuracy, precision, recall, and F1-score metrics.

**RQ2: What is the best performance of fully-trainable transfer learning for multiclass classification of UML diagrams?**

**Rationale:** The process of training all layers of neural networks takes more time and computation resources. The purpose of this RQ is to understand whether the cost of training all layers of neural networks will improve performance in the task of classifying UML diagrams. We evaluate the performance of fully-trainable transfer learning on the algorithms from RQ1 with all trainable layers.

**RQ3: Is transfer learning essential for multiclass classification of UML diagrams?**

**Rationale:** Transfer learning can speed up the process of training algorithms and can improve the accuracy of classification. However, images from the most popular dataset for transfer learning (ImageNet [30]) are not like UML diagrams from our dataset. This RQ aims to understand whether transfer
learning is better for classifying UML diagrams than training neural networks from scratch. We measure the performance of MobileNet and our proposed neural network without using transfer learning, and compare them with the best results from RQ1 and RQ2.

RQ4: What is the best performance on time per image of neural networks for multiclass classification of UML diagrams?

Rationale: The aim of this RQ is to investigate the performance of classification on time per image based on various neural networks. Since image classification can be used to collect datasets, refine search results, etc., the use of a classification algorithm should not be too time-consuming. The performance time is measured by using the GPU.

A. Data Collection

For the experiments, we created a dataset based on several existing datasets [10] [31] [9] [26], in which the Lindholmen dataset [10] is the largest one. We scrapped more than 10000 images from these datasets, and we then manually removed the duplicates and labeled four types of UML diagrams: class diagrams, activity diagrams, use case diagrams, sequence diagrams. Non-UML images were collected from [31] and manually filtered to remove UML diagrams. Overall we collected 3231 images (including 700 class diagrams, 454 activity diagrams, 651 use case diagrams, 706 sequence diagrams, and 720 non-UML images). Our dataset has been provided online for replication and reproduction purposes [32].

B. Image Classification Process

The process of classifying UML diagrams is composed of four phases:

Phase 1: Input Data. The input is the dataset which contains 700 class diagrams, 454 activity diagrams, 651 use case diagrams, 706 sequence diagrams, and 720 non-UML images, and we further split the dataset into a training set, a testing set, and a validation set.

Phase 2: Preprocess Images. The images were converted to a JPG format and to a size of 224x224 or 299x299.

Phase 3: Train Classification Algorithms. We trained different pre-trained neural networks and some neural networks without pre-trained weights.

Phase 4: Evaluate Trained Classification Algorithms. We evaluated the performance of each algorithm from Phase 3 using multiple performance measures.

In Phase 1 (i.e., input data), we split the dataset into validation, testing and training sets: 10% of images (323) as the validation set, 20% of images (646) as the testing set, and 70% of images (2262) as the training set. Deep learning frameworks often do not work with all image formats, so we converted all images to a JPG format. Also, for pre-trained models, we needed to bring all the images to the particular size, in this case, we brought the images to the size of 299x299x3 for InceptionV3 and 224x224x3 for the rest of the neural networks that we used (recommended image sizes for transfer learning). All the images were normalized by changing the range of pixel intensity values between 0 to 1. The training dataset was augmented with a horizontal flip, slight shifts in the horizontal and vertical axis (up to 20%).

To raise the accuracy and decrease the training time in Phase 3, we trained neural networks with and without using transfer learning, which is a method to reuse the information received during training on one task to new tasks. We used models pre-trained on the ImageNet task [30], including MobileNet, DenseNet169, NasNetMobile, ResNet152V2, and InceptionV3. Convolutional layers were not trained during the semi-trainable transfer learning process and all layers were trained during the fully-trainable transfer learning process. The output from pre-trained models was fed to a fully-connected layer with 1024 neurons and the relu activation function, then through dropout layer to the next fully-connected layer with 512 neurons and the relu activation function, and after the last dropout layer to a fully-connected layer with five neurons and a softmax activation function.

Our proposed neural network was inspired by MobileNetV3 [33] and ResNet [22]. Figure 1 provides the details about our proposed neural network architecture. The input layer is a convolutional layer with kernel size 3x3, 32 filters and ReLU as an activation non-linear function. Next are the architecture blocks, which are repeated throughout the architecture. The input of each logical block is a convolutional layer with twice as many filters as the previous layer and ReLU as an activation function. The next layer is the batch normalization layer. The output of the batch normalization layer is then split into two single outputs, one of which is called a shortcut which goes directly to the end of the logic block. Shortcut was made by using a maximum pooling layer. The second output goes through a separable convolution layer with half as many filters as the input layer and a 1x1 kernel size with ReLU activation function. It then goes into a convolution layer identical to the input layer of the logic block with stride 2 and the HardSwish activation function. Then goes to a similar layer but with LeakyReLU activation function. And after the batch normalization layer is concatenated with the shortcut output to the next logical block. Each logic block of our architecture contains convolution layers with different kernel size and stride, separable convolution layer [34]. We also used ReLU, LeakyReLU, and HardSwish [35] as activation functions. Following the sequential use of several logical blocks, the Global average pooling layer is used. Instead of using classical fully-connected layers at the end of the neural network, we used two convolutional layers with 1x1 kernel and the numbers of filters are 1024 and 5 (i.e., number of classes in the dataset). This allows us to reduce the number of parameters and speed up the neural network.

C. Performance Evaluation

We measured the performance of each algorithm by using accuracy, precision, recall, and F1-score, which are usually used in performance evaluation of image classification. The images from one class that have been correctly classified by a classifier are considered as True Positive (TP), while the
TABLE I: Performance in precision, recall, and F1-score by using semi-trainable transfer learning

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Recall</th>
<th>Precision</th>
<th>Accuracy</th>
<th>F1-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet152V2</td>
<td>88.28%</td>
<td>91.87%</td>
<td>96.09%</td>
<td>90.04%</td>
</tr>
<tr>
<td>InceptionV3</td>
<td>89.52%</td>
<td>91.88%</td>
<td>96.32%</td>
<td>90.69%</td>
</tr>
<tr>
<td>MobileNet</td>
<td>91.06%</td>
<td>92.77%</td>
<td>96.79%</td>
<td>91.91%</td>
</tr>
<tr>
<td>DenseNet169</td>
<td>89.41%</td>
<td>92.26%</td>
<td>96.38%</td>
<td>90.81%</td>
</tr>
<tr>
<td>NasNetMobile</td>
<td>85.41%</td>
<td>89.26%</td>
<td>95.03%</td>
<td>87.29%</td>
</tr>
</tbody>
</table>

To measure the time spent, we used the free version of Google Colaboratory. This will make the results reproducible and verifiable. The free version of Google Colaboratory uses Nvidia T4 as the GPU device.

IV. RESULTS AND ANALYSIS

To answer the RQs, we used deep learning algorithms. Overall, we experimented with 12 (5 classification algorithms as a semi-trainable transfer learning + 5 classification algorithms as a fully-trainable transfer learning + 2 classification algorithms without pre-trained weights) experiment configurations. These configurations were applied on a training set of 2262 images, a testing set of 646 images, and a validation set of 323 images. Since the neural networks have some bias in performance results, each experiment configuration was evaluated 5 times and the results below are an average of the 5 times when the experiment configuration is repeated.

RQ1: What is the best performance of semi-trainable transfer learning for classification of UML diagrams?

To answer RQ1, we applied several pre-trained neural networks such as MobileNet, DenseNet169, NasNetMobile, ResNet152V2, and InceptionV3 without training convolution layers. Table I shows the precision, recall, and F1-score for each algorithm. The best performed algorithm was MobileNet (accuracy = 96.79%, precision = 92.77%, recall = 91.06%, and F1-score = 91.91%). In the five classification algorithms, MobileNet outperforms other algorithms.

RQ2: What is the best performance of fully-trainable transfer learning for classification of UML diagrams?

To answer RQ2, we applied all pre-trained algorithms from RQ2 with training all layers. Table II shows the precision, recall, and F1-score for each algorithm. The best performed algorithm was DenseNet169 (accuracy = 97.76%, precision = 94.44%, recall = 94.35%, and F1-score = 94.40%). In the five classification algorithms, DenseNet169 outperforms other algorithms. Moreover, DenseNet169 with all trainable layers outperforms all semi-trainable algorithms from RQ1. However, the training process for all layers takes more time than the training process of semi-trainable transfer learning.

RQ3: Is transfer learning essential for multiclass classification of UML diagrams?

Since the images from the ImageNet dataset [30] used for transfer learning are not similar to the images from our set from the whole observations. Recall is the fraction of all images of one class correctly identified as True Negative (TN).

Accuracy (ACC) is the proportion of accurately predicted images from the whole observations. Precision is the ratio of images correctly identified as one class to all images identified as this class. Recall is the fraction of all images of one class correctly demarcated. The harmonic mean of precision and recall is called F1-score or F1 Measure. We calculated those metrics using the following equations:

\[
\text{ACC} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}
\]

\[
F1 - \text{score} = \frac{2TP}{2TP + FP + FN} \tag{2}
\]

\[
\text{precision} = \frac{TP}{TP + FP} \tag{3}
\]

\[
\text{recall} = \frac{TP}{TP + FN} \tag{4}
\]
dataset, the question arises about the need for transfer learning. To answer RQ3, we compared the best results from RQ1 (MobileNet (STTL)) and RQ2 (DenseNet169 (FTTL)) with the results of MobileNet and our proposed neural network architecture, which were trained without using transfer learning (WPW). The comparison shows that the pre-trained out-of-the-shelf neural networks are better than the off-the-shelf neural networks that are trained without using transfer learning, but worse than our proposed neural network at classifying UML diagrams. Table III shows the comparison between the pre-trained and not pre-trained neural networks. Moreover, it is possible to pick up a custom neural network architecture that will cope almost as well as off-the-shelf neural networks, but the process of picking architecture parameters takes time.

| TABLE II: Performance in precision, recall, and F1-score by using fully-trainable transfer learning |
|-------------------------------------------------|-------------------------------|-------------------------------|-----------------|-----------------|
| Recall                                          | Precision                     | Accuracy                      | F1-score        |
| ResNet152V2                                     | 91.11%                        | 93.50%                        | 97.33%          | 93.21%          |
| InceptionV3                                     | 94.14%                        | 94.24%                        | 97.68%          | 94.19%          |
| MobileNet                                       | 93.83%                        | 94.22%                        | 97.62%          | 94.03%          |
| DenseNet169                                     | 94.35%                        | 94.44%                        | 97.76%          | 94.40%          |
| NasNetMobile                                    | 87.15%                        | 90.21%                        | 95.54%          | 88.65%          |

RQ4: What is the best performance on time per image of neural networks for multiclass classification of UML diagrams?

We measured time per image by using Google Colaboratory to answer RQ4. Table IV shows the measured time for our approach and comparison with the solution from [9]. For the transfer learning approach, MobileNet is the most effective algorithm in term of performance time (0.014 second per image). For all the approaches, our proposed neural network showed the best performance time (0.0135 second per image).

| TABLE III: Comparison between pre-trained and not pre-trained neural networks (STTL = semi-trainable transfer learning, FTTL = fully-trainable transfer learning, WPW = training without pre-trained weights) |
|-------------------------------------------------|-------------------------------|-------------------------------|-----------------|-----------------|
| Recall                                          | Precision                     | Accuracy                      | F1-score        |
| MobileNet (STTL)                                | 91.06%                        | 92.77%                        | 96.79%          | 91.91%          |
| DenseNet169 (FTTL)                              | 94.35%                        | 94.44%                        | 97.76%          | 94.40%          |
| MobileNet (WPW)                                 | 88.59%                        | 89.23%                        | 95.58%          | 88.90%          |
| Our Solution (WPW)                              | 96.48%                        | 96.76%                        | 98.65%          | 96.62%          |

Summing up the results of the RQs, we can say that the DenseNet169 with all trainable layers is a good and out-of-the-shelf choice to classify UML diagrams from images. We have also found that it is possible to find an architecture that performs better than the pre-trained out-of-the-shelf neural networks, but that architecture selection takes time. Our proposed neural network architecture showed the best results (accuracy = 98.65%, precision = 96.76%, recall = 96.48%, and F1-score = 96.62%) and it also has the least parameters (2.4millions) and spends the least time per image (0.0135 seconds per image using GPU) for classifying UML diagrams.

V. Threats to Validity

Internal validity concentrates on whether the results can be drawn from the data, and one of the threats to the internal validity of this study are the overfitting of neural networks and the manual labelling of the dataset used in the experiment. Since pre-trained models are trained on a complex multi-class task of 1000 classes, they can be retrained when we move on to five class classification. To avoid overfitting, we used a Dropout layer after each fully connected layer. This allows us to ignore some of the information coming from fully connected layers and increases the stability of models. The images used in the experiment were manually labelled, which may introduce selection bias. It is possible that images were incorrectly labelled, leading to incorrect classification results. To mitigate this threat, the labeled data was manually checked by the first author, and for any unsure labels the first author discussed with the second author to get a consensus.

External validity reflects the extent to which the study results and findings can be generalized in other cases with similar characteristics. The potential threat to the external validity is about the diversity of the dataset used in the experiment, which was created based on several existing datasets [10] [31] [9] [26], in which the Lindholm dataset [10] is the largest UML models repository from OSS projects and the other three datasets were retrieved by Google Images search. We believe that the experiment results can be applicable to the classification of UML diagrams in practice to a large extent.

Construct validity in this work focuses on whether the evaluation metrics are suitable and measured correctly. A set of metrics (i.e., accuracy, recall, precision, and F1-score) were used to measure the performance of the classification algorithms, which have been widely used in assessing the quality of algorithms in image classification.

Reliability refers to whether the experiment yields the same results when it is replicated. In this work, this validity is mainly related to the dataset and the execution of the experiment. We defined the protocol for the classification process and evaluation metrics, which were confirmed and followed by all the researchers. We also made our experiment dataset available for replication purposes [32].

VI. Conclusions and Future Work

In this work, we automatically identify four most popular types of UML diagrams (class diagrams, use case diagrams, activity diagrams, and sequence diagrams) and non-UML
diagrams from images by using five popular neural network architectures (including MobileNet, DenseNet, NasNet, ResNet, and Inception) using transfer learning. We scraped over 10000 images from open datasets and manually labelled it into four types of UML diagrams. In total, we have collected a dataset that contains 3231 images (700 class diagrams, 454 activity diagrams, 651 use case diagrams, 706 sequence diagrams, and 720 non-UML images). Then, we used transfer learning to classify UML diagrams by applying the neural network architectures. We also proposed our neural network architecture for multiclass classification of UML diagrams. The experiment results show that our proposed neural network architecture outperformed the existing neural network architectures with an accuracy of 98.65%, a precision of 96.76%, a recall of 96.48%, and F1-score of 96.62%. To measure the time spent for the classification, we used a free version of Google Colaboratory. We found that the most efficient architecture for classifying UML diagrams and our proposed neural network architecture also has the least parameters (around 2.4 millions).

In the next step, we plan to (1) construct cost-effective neural network architectures for classifying major types of UML diagrams from images; and (2) automatically recover the relationships between various types of UML diagrams from images within a project in order to establish the tractability between UML models.

REFERENCES


On Integrating Ethicality in User Stories

Pankaj Kamthan  
Computer Science and Software Engineering  
Concordia University  
Montreal, Canada  
pankaj.kamthan@concordia.ca  

Nazlie Shahmir  
Information Services Governance  
Canadian Pacific Railway Limited  
Calgary, Canada  
nazlie_shahmir@cpr.ca  

Abstract—In recent years, software has increasingly become anthropomorphic, even autocratic. For example, software is being used exclusively for activities, such as decision-making, question-answering, or recommending, that in the past were either partly or entirely human. This has only contributed to the enduring issue of software ethics. In that regard, this paper models ethicality as a meta-quality attribute and proposes an ethically-sensitive, standards-based, technology-and-tool-independent, semi-formal framework, comprising interrelated conceptual (meta-)models that provide an understanding of ethicality, user story environment, and user story process. It outlines an approach of integrating ethicality naturally and systematically in the user story process, and provides illustrative and representative examples in support of this approach. Finally, it presents the results of a preliminary survey of students and professionals on their knowledge and experience of ethics in (agile) software projects.

Keywords-axiology; conceptual modeling; ethical dilemma; human-centered agile methodology; interactive system; software quality

I. INTRODUCTION

The history of computer ethics predates that of software engineering [1, 2]. Fig. 1 gives an approximate timeline of different types of ethics that have been a subject of attention in relation to computing, as the role of computing itself evolved based on the needs of the society. However, increasing 'softwareization' of a variety of application domains, along with essentially uncontrolled and unlimited malleability of software, including those for unscrupulous or maleficient purposes, has made the issue of software ethics as exigent as ever. The consequences have ranged from innocuous and reversible, albeit at the cost of time and effort, to extremely nocuous and irreversible [3]. This situation is clearly untenable.

II. BACKGROUND AND RELATED WORK

A. Nature of Ethicality from a Software Engineering Perspective

This paper distinguishes among actions or inactions that are prudential, ethical, and legal [5]. Fig. 2 illustrates the interrelationship between these concepts by means of a Venn Diagram. An action or inaction is prudential if it is in a person’s interest; an action or inaction is legal if it is not explicitly prohibited by law of a jurisdiction; and an action or inaction is ethical if it does not violate certain (personal, organizational, and/or societal) codes of ethics. Furthermore, ethics could be either deontological (an action or inaction is ethical or unethical in itself) or teleological (an action or inaction is ethical or unethical depending on its consequences).

DOI reference number: 10.18293/SEKE2021-001

Figure 1. A panorama of different types of ethics over different decades.

Figure 2. The ethical, legal, or prudential actions or inactions in context.
For example, while booking a flight it may be prudent for a traveler to have travel insurance, but it is teleologically unethical for a flight reservation system to add the cost of travel insurance to the cost of the flight automatically without the traveler’s consent, and it may be illegal for the traveler to try to get travel insurance by providing incorrect, inconsistent, or incomplete information.

From the perspective of applicability, ethics could be classified into macroethics (large-scale, generic, coarser, and applies to organizations) and microethics (small-scale, specific, granular, and applies to individuals) [6]. The two are complementary, usually coexist, and are necessary for the practice of software engineering. It is possible to have one without the other. For example, the management of a software development company may engage in so-called ‘ethics washing’, but its requirements engineers may still act ethically.

B. Previous Work on Ethics in Software Requirements Engineering

The interest in integrating ethics in software RE and related areas is relatively recent and, in part, motivation for this paper. In [7], scenarios of unethical practices and their negative impacts on the users are given, and in [8] the need for requirements to reflect “socially responsibility” is underscored. To increase awareness of the ethical implications of software from a RE perspective [9], a systematic literature review and grey literature review was conducted in [10], and as number of ethical issues are highlighted. The ACM/IEEE Software Engineering Code of Ethics and Professional Practice (SECEPP) lists generic principles, each of which is refined into specific clauses, related to the behavior of and decisions made by professional software engineers as well as students of the profession. Finally, the IEEE Standard 7010 provides guidelines for Ethically Aligned Design (EAD).

III. AN OVERVIEW OF A CONCEPTUAL FRAMEWORK FOR ETHICALITY IN USER STORIES

A. Ethicality as a Meta-Quality Attribute

A meta-quality attribute is a quality attribute about quality attributes. In this sense, ethicality is an anthropomorphic, extrinsic, meta-quality attribute, aiming to mimic certain aspects of sentience considered much desirable among humans.

There can be degrees of ethicality as an extrinsic property. For example, the severity of ethicality is especially acute in mission-critical applications, such as those that are high-risk safety-, privacy-, or security-critical, as opposed to low-risk casual applications.

As per model-based software quality engineering [11, 12], it is acknowledged that the notion of ethicality in and of itself is at a rather high a level to be useful, and therefore it needs to be decomposed into multiple, low levels to be meaningfully practical. This decomposition lends a hierarchical structure, as shown in Fig. 3. The mapping between extrinsic and intrinsic properties, as well as between intrinsic properties and entities of knowledge, is many-to-many (as evident also by Table 1). The knowledge entities could, for example, include principles, guidelines, patterns, and metrics.

For example, identifiability (intrinsic quality attribute) contributes to traceability (extrinsic quality attribute) that, in turn, contributes to transparency (extrinsic quality attribute), and that, again, in turn, contributes to ethicality.

B. A Conceptual Meta-Model for User Story Environment

Fig. 4 shows a conceptual meta-model of a part of the user story environment from a managerial perspective. A user story is an aggregation of role, goal, and value, in that spatial order, and is associated with a priority and acceptance criteria. A user story together with its acceptance criteria is used to estimate the time and effort needed for completing it. The priority and estimate of a user story are based on the risk associated with that user story, which is of concern to both the users and the product owner [13, 14].

C. The Outline of an Ethically-Sensitive User Story Process

It is understood that certain desirable external quality attributes, such as safety, privacy, and security, cannot be addressed properly, if at all, at the end of development. It is therefore important that ethicality be considered at the beginning of a software development process, and remains an explicit concern throughout all stages of development.
Fig. 5 shows a conceptual meta-model for a minimal, continuous user story process [15], the activities and artifacts of which are aimed to be ethically-sensitive.

![Conceptual Meta-Model for Ethically-Sensitive User Story Process](image)

Figure 5. A conceptual meta-model for ethically-sensitive user story process.

The user story process proceeds as follows. The relevant stakeholders participate in ethics poker, following which the user stories are prioritized based on risk, and user stories that pose high risk with respect to ethicality are placed at high priority. The result of this step is a collection of ethicality stories for the current iteration. This is followed by the creation of a value-centered [16] and responsibility-driven [17] prototype for the ethicality stories by following the principles and patterns of Design Thinking [18] and Systems Thinking [19]. This experimental prototype could after a demonstration prompt a refinement of the user stories and possibly the elicitation of new user stories. Finally, there is an evaluation of the prototype involving actual users, for example, through crowdsourcing, observation, and/or survey, subject to informed consent. This could lead to feedback about ethicality of the product (which could prompt a refinement of the user stories and/or the prototype), as well as about the process (which could prompt an improvement of the user story process).

Table 1 lists a compendium of classical as well as novel quality attributes that are necessary for ethicality as identified by a number of recent studies [8-10, 20-22], and examples of corresponding user stories for a variety of application domains. It could be noted that the mapping between the set of quality attributes and the set of user story examples is many-to-many.

The challenges to the user story process include being able to elicit tacit or implicit knowledge from potential users, to mitigate cognitive biases (such as Representativeness Bias), and to be aware of ethical dilemmas (such as Mission Impossible) [23], and being able to control the accrual of user story debt, a type of technical debt.

**TABLE 1. EXAMPLES OF USER STORIES FOR ETHICALITY**

<table>
<thead>
<tr>
<th>Privacy, Security, Well-Being</th>
<th>US9: A customer can contact the administrator of the shopping system about the item return policy to be able to shop with surety and serenity.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>US10: A reader can distinguish between content and advertisement on the media portal to be able to discern accordingly.</td>
</tr>
<tr>
<td>Axiology, Fairness, Utility</td>
<td>US11: A patron with achromatopsia can navigate through the library system to be able to seek the books of his or her choice.</td>
</tr>
<tr>
<td></td>
<td>US12: A student can report the presence of a malfunctioning critical user interface element on the course registration system so as to save time and effort of other students.</td>
</tr>
<tr>
<td>Accessibility, Sustainability, Well-Being</td>
<td>US13: A project manager can check on the enterprise information system the daily calendar of all team members to be able to monitor their engagements.</td>
</tr>
<tr>
<td></td>
<td>US14: A maintainer can see on the source code management system the rationale associated with the status of (say, accepting or rejecting) a defect to be assured that the defect management process is being followed properly.</td>
</tr>
<tr>
<td></td>
<td>US15: An auditor can independently access data used for performance benchmarks of a simulation system so that he or she can be assured of the correctness of the results and the consistency of the claims.</td>
</tr>
<tr>
<td>Accountability, Explainability, Traceability, Transparency</td>
<td>US16: A programmer can independently access on the source code management system the review checklist against which his or her source code was reviewed so as to improve his or her programming capabilities.</td>
</tr>
<tr>
<td>Competency, Traceability</td>
<td>US17: A programmer can independently access on the source code management system the review checklist against which his or her source code was reviewed so as to improve his or her programming capabilities.</td>
</tr>
</tbody>
</table>

**D. A Survey on Software Engineering Ethics**

To better understand the current state of knowledge and experience of software engineering ethics, including its relation to software quality, by those in academia and industry, a small-scale survey was conducted between Winter 2019 and Winter 2021. The respondents were from Canada, and consisted of graduate students in computer science or software engineering programs, and professionals in multiple software-intensive organizations, some of whom had been exposed to software engineering ethics. The survey had 16 items, each based on a 5-point Likert Scale ranging from Strongly Disagree (1) to Strongly Agree (5). Fig. 6 shows the results of 20 complete responses. It can be concluded from Fig. 6 that a majority of respondents understood the noxious (I6) and innocuous (I11) impact of unethical behavior, and believed ethical behavior could change over time through guidance and training (I16), but there was no agreement on the responses on the issue of whether software engineering waste is unethical (I15).

![Distribution of Selected Responses](image)

Figure 6. The distribution of selected responses from an ethics survey.
IV. DIRECTIONS FOR FUTURE RESEARCH

A. “Who Dunnit?”: Causal Analysis of Violations of Ethicality

There can be a number of underlying causes of violations of ethicality: scarcity of resources, inadequate education in the application domain or ethicality engineering, inadequate elicitation of user needs, short-term expediency due to the pressure of time-to-market [24], lack of foresight, presence of (meta-)cognitive biases such as the Dunning-Kruger Effect, resorting to logical fallacies, gender inequity, politics, or deficiency of soft skills necessary for interviewing, negotiating, or reporting. Indeed, knowing the origins (or root causes) of such violations could be useful for a preventive approach towards ethicality, and is therefore of research interest.

B. “Get ’em Early”: Ethicality in Requirements Engineering Education

The education that the students receive as learners of today creates, directs, and shapes their attitudes, habits, and temperaments as practitioners or researchers of tomorrow and beyond. For these traits to be socially-acceptable, ethics needs to be, as with the user story process, introduced as early as possible in the RE curriculum and emphasized throughout by lessons from history of software ethics, examples of ethical dilemmas, and case studies of ethical violations as per at least the SECEPP along with their potentially adverse consequences for society-at-large [25]. Indeed, a strong commitment to ethicality needs be a part of lifelong learning of all the students, for society-at-large [25]. To investigate suitable approaches for doing so are therefore also of research interest.

V. CONCLUSION

The reasons for and the aspirations of software ethics are at least as relevant today as they were ~ 70 years ago. To be able to view ethicality as an extrinsic property of a software system lends itself to the established preventive as well as corrective knowledge in conceptual modeling, (agile) RE, and software quality engineering, as this paper has attempted to show.

The COVID-19 pandemic has led to a sobering reminder that a sole-effort of prevention or vaccination is insufficient, and that “it will take all of us”. In a similar vein, ethicality may be realized to a notable extent only if it is perceived, discerned, and approached as a collectively-shared responsibility by all those who impact or are impacted by software. Having in place policies, processes, and procedures for ethicality is useful, perhaps even necessary, but these instruments have their inevitable limits [13, 26], even with the best of intentions and executions. In the end, the society may have to learn to live with the degrees of ethicality of software, of its own making.

ACKNOWLEDGMENT

This paper is dedicated to the indelible memories of Terrill Fancott and Peter Grogono, who passed away in 2020 and 2021, respectively. The authors are grateful to CUPFA for a Professional Development Grant, and to responders of the survey for their interest and participation.

REFERENCES

A Collaborative Forensic Framework for Detecting Advanced Persistent Threats

Weifeng Xu  
*Forensic Science: Cyber Investigations*  
University of Baltimore  
Baltimore, USA  
w xu@ubalt.edu

Jie Yan  
*Department of Computer Science*  
Bowie State University  
Bowie, USA  
jyan@bowiestate.edu

Daryl Stone  
*Department of Technology and Security*  
Bowie State University  
Bowie, USA  
dstone@bowiestate.edu

**Abstract**—An advanced persistent threat (APT) is one type of cybercrime that steals valuable information over an extended period through malicious activities. The paper proposes a collaborative framework to systematically detect APTs by analyzing the Cyber Forensic Evidence (CFE) collected from a System Under Investigation (SUI). It is a post-compromise analysis based on Forensic-Evidence-Driven Finite State Machines (FED-FSM) modeled from a SUI. A FED-FSM extends an FSM by defining a set of forensic evidence patterns as guided conditions that trigger the state changes of FSM. The approach consists of three tasks (1) collecting shared CFE and formalizing patterns of CFE, (2) modeling the security status of an SUI in a FED-FSM, and (3) building a Threat Activity Detection Engine to match the observed CFE from SUI logs with the CFE patterns in the FED-FSM. An empirical study shows the framework can be used to detect malicious activities of Poison Ivy, which utilizes a remote access tool to control computers.

**Keywords**—advanced persistent threat, Structured Threat Information Expression, finite state machine, threat detection, ATT&CK framework, digital forensics

I. INTRODUCTION

An advanced persistent threat (APT) is one of the cybercrimes, which typically refers to a state-sponsored hacking group and cyberattacks associated with the group. APT strategizes its way to infiltrate an organization’s network and exfiltrates valuable information. For example, one common attack executed by APTs is to place custom malicious code on one or multiple computers for specific stealth tasks. Various approaches have been proposed to study and detect APT activities, either through scanning signatures of malicious code [1][2][3] or analyzing evidence that malicious programs are left on systems or networks activity logs [4] [5]. A common issue of these approaches is that they act alone during the whole investigation process: they set their environments, collect evidence, observe patterns, and make conclusions based on their judgments. It is hard for the cybersecurity community to recreate their environments, verify their observations, and reuse their observation results. MITRE [6][7] has proposed an Intelligence-sharing-based approach to fight against cybercrimes with collaborative efforts. It developed a concept called shared Cyber Threat Intelligence (CTI) to facilitate knowledge sharing. For example, Figure 1 shows a shared scenario which consists of two predefined objects indicator and malware. The figure means that a cybercrime investigator has reported that he/she has found malware, named Poison Ivy (PIVY). PIVY is a remote access tool used by many APTs and cybercriminals for information infiltration. To support his/her findings, the investigator attaches the hash code of the malware to the indicator object, which defines the possible threat patterns of malware, e.g., the SHA hash code of the PIVY executable file.

![Figure 1. A simple indicator uses a file hash to indicate the presence of Poison Ivy](image)

Shared CTI sounds promising for collaborative threat detection and therefore, has attracted much attention in cybersecurity communities. However, there are major challenges to apply them directly in practice due to: (1) Current CTI framework does not support presentable CTI in terms of forensic investigations. Detecting APT threats is the process of investigating cybercrimes with supporting digital evidence. A presentable shared CTI has to describe how threat evidence is acquired, preserved, identified, and validated to meet the requirements of the law, (2) the lack of a systematical approach to discover digital forensic evidence for a System Under Investigation (SUI). Current CTI only describes the evidence and the patterns of threat evidence using Indicator, it doesn’t provide practical guidelines of how evidence can be extracted for a given SUI, and (3) Current CTI cannot describe the dynamical behaviors of an APT. Current CTI is a threat data model, which only describes static objects. To detect APTs, a dynamical model is needed to describe the known behaviors of APTs so that the model can be used for matching a given SUI.

The paper demonstrates a framework to detect APTs imposed on an SUI over time using Forensic-Evidence-Driven Finite state machines (FED-FSM). The approach first enhances CTI to Cyber Forensic Intelligence (CFI) and enables indicator objects to capture the properties of digital forensic evidence. The enhanced indicator object is called Cyber Forensic Evidence (CFE) object. We then propose a systematic approach to extract CFE-objects from an SUI and specify the SUI as a collection of CFE objects. Finally, we model the SUI in a FED-FSM with CFE objects as the model’s guided conditions. We build a Threat Activity Detection Engine (TADE) to detect APTs that impose on the SUI by comparing the observed CFE objects with the...
CFE objects defined in FED-FSM. The contribution of the paper includes: (1) the formalization of CFE by leveraging ATT&CK knowledge model [6][7] (2) inferring dynamical behaviors of an APT using FED-FSM, and (3) designing a scalable threat-activity detection framework for detecting real-world APTs systematically.

The rest of the paper is outlined as follows: Sections II and III describe the difference between CTI and CFE and how CEF can be discovered systematically. Section IV shows the proposed architecture for detecting APT activities. Section V formally defines FED-FSM. Section VI describes an empirical study. Section VII summarizes the related work. Finally, Section VIII concludes this paper.

II. SHARED-CYBER THREAT INTELLIGENCE AND CYBER FORENSIC EVIDENCE

Intelligence-sharing is a critical strategy for cybersecurity defenders because it allows them to avoid the missteps of their peers within the security community and to deploy proven defensive measures. To systematically and effectively share security Intelligence, research groups [6][7] have realized that cybersecurity-related terminologies, measurements, and standards need to be defined to understand the security and share CTI. For example, to better understand security problems, MITRE has created an ATT&CK framework to document common Tactics, Techniques, and Procedures (TTP) based on real-world observations of adversaries’ operations against computer networks. To share CTI, OASIS CTI technical committee [8] has developed Structured Threat Information Expression (STIX) to formalize CTI and enable interoperable sharing of CTI across organizational, technology, and geographic boundaries. STIX is considered a de facto standard for many industries and organizations [9]. Figure 2 shows a shareable CTI expressed in STIX objects based on Fireeye's Poison Ivy (PIVY) Report [10]. Colored icons in the figure represent various types of objects defined in STIX. Each STIX object defines various attributes to specify the object. For example, the indicator object shown in Figure 1 has two attributes, the type of indicator, e.g., malware activity, and the pattern of the indicator, i.e., the hash code of the malware file. Note that the PIVY data model shown in Figure 2 primarily relies on 25 indicator objects (i.e., blue circles with fingerprint symbols) to represent suspicious or malicious cyber activities. These indicators are spread out all over the model and there are no pre-defined semantic relations among them.

We extend shared CTI to CFE to align with the general process model of digital forensic investigation, including evidence collection, analysis, interpretation, and validation. Figure 3 shows the CFE object extends the Indicator object in CTI. The hollow arrow represents the extended relationship and the solid diamonds indicate collection tool, media, investigator, etc. are components of forensic evidence object. We use CFE object and its components to answer the questions related to the legal aspects of digital forensics [11][12]:

- Where is forensic evidence collected, e.g., memory or disks? It can be addressed by Collection Tool and Media objects, which are used to describe the source of the evidence and how evidence is extracted and stored.
- Who does collect the evidence? When did a person access the original digital evidence? Is the person forensically competent? This information can be addressed by an investigator object.
- How does the analysis carry out? How to interpret the results? A solid forensic evidence analysis should be based on math and science. Besides the pattern match method defined in the Indicator object, we propose the Identification objects in CFE to record other methods

![Figure 2: A threat data model based on Fireeye's PIVY report](image)
and algorithms used for evidence analysis and explanation. For example, evidence patterns or new evidence can be discovered by artificial intelligence (AI).

- Most importantly, how the above process and results are validated? Why should courts trust the process and results? For example, upon seizing digital evidence, action should not change that evidence. We propose Preservation and Interpretation objects to validate the authenticity of evidence and explain the. For example, if evidence patterns or new evidence is discovered by artificial intelligence (AI), then explainable AI algorithms may show investigators and courts how much they can trust evidence.

![Diagram](image_url)

**Figure 3. CFE object extends Indicator object in CTI**

### III. Cyber Forensic Evidence Discovery

We propose a systematic approach to search for possible CFE for an SUI. The main idea of the approach is based on the fact that all CFE is generated by software and hardware of the SUI that was used by cybercriminals. Figure 4 shows a bottom-up layered architecture to discover CFE. Layer one represents any cybercrimes involving computing-related devices. Layer two is called a forensic evidence generator layer. It represents a computing-related device used by cybercriminals and is considered a forensic evidence generator. All system components of the device, such as software and hardware, will either generate CFE or store generated CFE. Layers three and four classify CFE objects in a tree-like structure. In the structure, all CFE is the root of the CFE tree, layer three represent CFE categories, and layer four contains all CFE objects. The first level of layer three has three types of CEF, including application-generated CFE, operating system (OS)-generated CFE, and hardware-generated CFE.

The application-generated CFE has two sub-types, the application function-generated CFE and application non-function-generated CFE. From the software engineering perspective, an application is developed based on two types of requirements, functional and non-functional requirements. A functional requirement is a description of the service that the software must offer. It defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs [13]. Non-functional requirements are often referred to as "quality attributes" of a system, including usability requirements, security requirements, reliability requirements, etc. For example, an online chatting application that is built from functional requirements will generate CFE objects related to the chat functions, such as chat text messages, images, audio, and videos. To meet the usability requirements, the online chat application may cache credentials in cookie or memory and the cookie is considered a classic CEF object. The security requirements may save the private key of the application in a folder. The key is a CFE too. These identified CFE objects are in layer four. Similarly, OS-generated CFO also can be classified into two sub-categories, OS-function and non-function generated CFE objects. A typical operating system has three main functions: (1) manage the computer's resources, such as the Central processing unit (CPU), memory, and disk drives, (2) provide a user interface, and (3) execute and provide services for applications software. The non-functional features of an OS can also have usability, security, reliability requirements, etc. Note that layer three can be further refined to form sub-types as needed.

![Diagram](image_url)

**Figure 4. A systematical approach to discover digital forensic evidence for a system under the investigations**

The hardware-generated CFE is different from the software-generated CFE, including the application- and OS-generated CFE. It contains three sub-types of CEF related to the hardware components of commuting devices, CPU-generated CFE, hard drive-generated CFE, and memory-generated CFE. Memory forensics is a vital form of cyber investigation that allows an investigator to identify unauthorized and even malicious activity on a target computer or server. This is usually achieved by running special software, known as a memory dump, which captures the current state of the system’s memory as a snapshot file. The memory-generated CFE presents some special states in memory dumps. There are two types of memory-generated CFE, volatile memory generated CFE and non-volatile memory generated CFE. Volatility Framework [14], a volatile memory extraction utility framework, can extract many CFE objects, such as clipboard, cmdline, and iehistory. The non-volatile
memory CFE refers to firmware generated CFE. Firmware is a specific class of computer software that provides low-level control for a device's specific hardware. Typical firmware CFE objects include Basic Input/Output System (BIOS), hard driver, and routers and firewall firmware. These CFE objects are in level four and the leaves of layer three.

Formally, we can model the relations between cybercrimes and their associated CFE.

- All committed cybercrimes \( C \) in layer one is a collection of crime \( C = \{ c_1, c_2, \ldots, c_n \} \), where each cybercrime \( c_i \in C \) and \( 1 \leq i \leq n \).
- All CFE objects \( E \) in layer four is a collection of CFE \( E = \{ e_1, e_2, \ldots, e_n \} \), where \( e_i \in E \) and \( 1 \leq i \leq n \).
- CFE generated from a crime \( c \) under the investigations is a set of CFE object \( V = g(c) \subset E \) where \( g \) is a CFE generation function, which represents how the crime \( c \) is committed.

IV. AN ARCHITECTURE FOR DETECTING APT CYBERCRIME ACTIVITIES

Figure 5 shows the architecture of detecting APT cybercrimes. The architecture has three main components shown inside of the dashed rectangle: a CFE model repository, a FED-FSM model repository, and a TADE. The system takes shared APT CFI, SUIs, and observable CFE objects from the SUI as inputs to infer possible malicious activities.

![Figure 5. The architecture for detecting APT Crime activities with FED-FSM](image)

A CFE Model Repository

A CFE model repository is a collection of sharable CFI from APTs expressed in JSON. Each CFE model consists of a list of CFE objects. Formally, let define \( c = \text{PIVY} \) and \( g \) is the function that generates all the CEF objects, i.e., \( V = g(c) = \{ \text{PIVY file, IP address, Running process, ... } \} \). PIVY file is a CFE object, which refers to the existence of the malware file itself (i.e., the static executable file). The IP address refers to the IP address of attackers, i.e., the client of PIVY is a CFE object because attackers need to communicate with the PIVY server. Also, the running process is another CFE object that indicates PIVY is running on victims’ devices.

The following JSON file defines the PIVY file CFE object. The CFE object defines CFE attributes that are associated with a threat activity. These attributes include the CFE type, id, name, pattern, etc. Patterns are designed to assert suspicious or malicious cyber activities. Specifically, patterns use observable objects and their attributes to describe forensic evidence that is associated with known malicious activities. For example, the pattern \{file: name = 'Poison_Ivy_2.3.2.exe'\} is to assert the existence of the PIVY file. Logic operations can be applied to multiple observable objects as well. For example, in addition to asserting the existence of the specific file, the following pattern can check the identity of the file (e.g., an SHA-256 hash) with the logical operator AND, \{file: name = 'Poison_Ivy_2.3.2.exe' AND file: hashes='SHA-256' = '...e9f5'\}. Note that STIX defines a cyber-observable object dictionary. Indicators containing cyber observable objects can be collected from threat data model repositories available publicly or created to support the flexibility of the framework.

![Figure 6. Code Snippet of a CFE object in PIVY](image)

B. A FED-FSM Model Repository

A FED-FSM model repository contains a collection of FED-FSM models. The main design idea of the repository is that (1) FED-FSM models in the repository describe potential APT activities imposed on SUIs and (2) any CFI we have observed in SUIs to infer and monitor the potential APT activities in terms of FED-FSM models can be used.

Traditional FSM is a well-studied mathematical model of computation, and these mathematical models are suitable for process automation. Unlike static threat data models for CTI sharing, which only describe static threat information, FSM is commonly used for capturing dynamical behaviors of synchronous sequential machines or software systems, and it has been utilized for detecting security vulnerabilities [15][16][17][18]. A state in FSM models is a description of the status of a system. An FSM model often contains a list of its
states and one initial state. A state of an FSM model can change from one state to another in response to some activities or external events. Such a change is called a transition.

However, traditional FSM models cannot be used for SUI threat detection directly because their transitions, such as attacking or threat activities, are unknown or unpredictable for threat analysis. Without knowing these threat activities, it will be very challenging for analysts to monitor and understand the status of SUI and to detect threat activities. To address the issue, we extend FSM by only allowing FSM states to change in response to forensic evidence left by APTs and patterned CFE objects. These CFE objects are from the CFE model repository. The status of a FED-FSM model is inferred by CFE objects instead of triggered by unknown threat activities. The formal definition of FED-FSM is discussed in the next section for process automation.

C. A Threat Activity Detection Engine

TADE shown in Figure 7 is another key component of the APT detection system. TADE consists of three different data types and threat activity detecting algorithms. These data types include CFE, cyber observed data, and sightings. The idea of the TADE is to use algorithms to infer the existence of malicious based on observed forensic evidence CFE extracted from an SUI and shared CFE fed on other cybersecurity intelligence resources, such as Anomali [19].

![Figure 7. The APT activity detection engine (TADE) for detecting threat activities using CFE (e.g., Indicator). Cyber observed data, and Sightings](image)

The detection engine has defined the following major functions:

- Collecting evidence logs. The log files include APT activity forensic evidence collected from files, disks, networking, and host system events as well as processes and signature strings in memory (Figure 4).
- Preprocessing logs. Logs will be cleaned, checked, organized, integrated, and stored in an evidence data repository or in memory for better performance.
- Formalizing observed evidence data. Similarly, the observed evidence data collected from an SUI will be specified in supporting STIX Domain Objects (SDO), named Observed Object or Observed CFE objects. Note that while CFE objects represent intelligence assertions behind attacks, raw observed information helps formulate the basis behind this intelligence, the observed CFE objects convey information that was observed on systems and networks. Multiple observed CFE objects can be used for crosschecking evidence and therefore increase the confidence of inferring results. The following code snippet shows an observed CFE file object, including its name, size, hash codes, etc.

```json
# An observed file object in PIVY
{
  "type": "observed CFE Object",
  "id": "observed-data--1",
  "objects": {
    0: {
      "type": "file",
      "hashes": {
        "MD5": "CF7AB60B7948232C447F284FC695A8689f",
        "SHA-256": "6cd85b47806e479d8f9f19be9f5",
        "Poison_Ivy_2.3.2.exe",
        "size": 54824
      }
    }
  }
}
```

![Figure 8. Code Snippet of an observed file object in PIVY](image)

- Determining threats. To detect a threat activity, we first use the STIX Relationship Object (SRO), i.e., Sighting object, to report observations of both CFE objects and observed CFE objects. Sighting objects use two references to capture: what indicator was sighted (i.e., sighting_of_ref) and what was seen on an SUI, (i.e., observed_data_ref). SROs are also specified in JSON to facilitate the threat detection automation process. Based on information collected by Sighting objects, various detecting algorithms can be used for determining whether threat activities exist by using patterns in CFE objects against observed data attributes.

V. FORENSIC-EVIDENCE-DRIVEN FSM MODEL

A FED-FSM model extends an FSM model by integrating a CFE model into the FSM model. Specifically, transitions of an FSM are determined by threat indicators specified by CFE objects. Formally, a FED-FSM model is defined as a tuple <S, T, F, I, L, q, s0>, where the elements of the tuple are defined as follows:

1. S is a set of states of an SUI.
2. T is a set of transitions of an SUI.
3. F is a finite set of arcs from one transition to another, i.e., \( F \subseteq S \times S \).
4. I is a set of threat indicators specified by CFE objects.
5. L is a threat indicators-selecting function on T and I, i.e., \( L(t, I) \subseteq I \) and \( t \in T \).
6. \( q \) is a guard function on T and L. The guard condition of transition t, \( q(t, L) \), is a first-order logical formula, which can be evaluated as true or false. The element of the formula is a list of STIX patterns that represent CFE objects.
7. \( s_0 \) is an initial state. It is often defined as Secure, i.e., \( s_0 = \text{Secure} \) and \( s_0 \in T \).

Figure 9 shows two states (i.e., Secure and Penetrated) and one transition of the PIVY FED-FSM model based on Fireeye’s PIVY report [10]. The state Secure is an initial state and it...
indicates a system has not been compromised. Penetrated state indicates malicious code that has been successfully executed on an SUI by an attacker. The tuple \( t = (\text{Secure, Penetrated}) \) is a transition. The threat indicator-selecting function \( L(t, I) \) selects a CFE object with a process ID, e.g., \( \text{observed-data} - 2 \), from \( I \), where \( I \) represents all available CFE objects in the CFE model repository. The guard function \( \varphi(t, L) \) on the transition \( t \) defines the pattern formally, i.e.,

\[
\text{[ pattern":"[process:name=myPoisonIvy_HK"]" ]}
\]

TADE will use the pattern to evaluate observed CEF objects collected from an SUI and return true if the pattern matches observed CEF objects or false if it doesn’t. The Boolean value determines whether the current state will change from Secure to Penetrated.

Figure 9. Two states and one transition from Poison Ivy FED-FSM model

FED-FSM models can also be expressed in Amazon States Language (ASL) in JSON format [20]. The following code snippet shows two states, Secure and Penetrated, as well as the lambda pseudo function (called resource), DetectProcess, for determining whether a threat exists in an SUI.

```json
1    "Comment": "A partial code snippet of Poison Ivy FED-FSM model in ASL",
2    "StartAt": "Secure",
3    "States": {  
4      "Secure": {  
5        "Type": "Task",
6        "Resource": "DetectProcess",
7        "Next": "Penetrated"
8      },
9      "Penetrated": {...}
10    }
11  
12  # DetectProcess lambda pseudo
13  # function as a transition
14  exports.handler =
15  function(event, context) {
16    context.succeed(
17      indicator.match({
18        observedData})
19  });
20}
```

VI. EMPIRICAL STUDY

The empirical study demonstrates the use of the framework to detect an APT that utilizes a customized PIVY against an SUI. We describe the following three artifacts related to the case study.

A. Case Study Environment Setting

The case study is conducted in a VirtualBox with two Windows virtual machines (VMs) and one security onion (https://securityonionsolutions.com/) Linux VM. One Windows VM acts as a PIVY client and another one acts as a PIVY server. The security onion is to monitor network traffics among three VMs. The client that is controlled by an attacker is configured on the attacker’s machine. It will accept the server’s connection and act as a command and control center of the server. The server or payload is created by the attacker using Poison Ivy 2.3.2 and then distributed to one victim’s machine. Once the victim executes the payload, the payload will infect its machine and connect to the computer running the PIVY Client. Malicious activities, including internal reconnaissance and data exfiltration [21], will be carried out after the victim’s machine has been infected. The environment setting instructions for the empirical study can be accessed at GitHub [22]. Figure 10 shows the PIVY client and a victim’s machine that has been infected by a PIVY server. The PIVY client is listening on its port 3460.

B. PIVY FED-FSM Model

Figure 11 shows the PIVY FED-FSM model of an SUI. Besides the two aforementioned states, Secure and Penetrated, the FED-FSM model has two states: Explored and Exfiltrated. The state Explored indicates an SUI has been explored by attackers to gain a better understanding of the environment for future actions. The state Exfiltrated indicates the SUI has an unauthorized movement of data.

The model has three types of guard functions on ten transitions. Each guard function contains multiple comparison expressions. For example, the guard function \( \varphi 3 \) indicates that at any state except Secure, the destination state will be Exfiltrated if both comparison expressions \( c8 \) and \( c9 \) are evaluated as true.

Table 1 lists ten representative comparison expressions used in the model for matching possible threat evidence collected from logs. The types of evidence include file, directory, process, Windows registry, IP address, and network traffic.
Table 1. Ten comparison expressions used in the FED-FSM model

<table>
<thead>
<tr>
<th>ID</th>
<th>Comparison Expressions</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td>file: name = 'myPoisonIvy HK.exe'</td>
<td>the name of PIVY Server</td>
</tr>
<tr>
<td>c</td>
<td>file: name MATCHES 'BG.bat.lnk'</td>
<td>a created file</td>
</tr>
<tr>
<td>c</td>
<td>Directory: path LIKE 'c:\Windows\%%\dfed'</td>
<td>a created folder</td>
</tr>
<tr>
<td>c</td>
<td>Process: name = 'myPoisonIvy HK'</td>
<td>a running process</td>
</tr>
<tr>
<td>c</td>
<td>win-registry-key: key = '\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\myPoisonIvy autorun'</td>
<td>a created registry key for autorun</td>
</tr>
<tr>
<td>c</td>
<td>ipv4-addr: value='10.0.2.12'</td>
<td>an IP address</td>
</tr>
<tr>
<td>c</td>
<td>ipv4-addr: value='10.0.2.12/24'</td>
<td>a subset of IP address</td>
</tr>
<tr>
<td>c</td>
<td>network-traffic:dst_ref.value = '10.0.2.12'</td>
<td>network traffic to IP '10.0.2.12'</td>
</tr>
<tr>
<td>c</td>
<td>network-traffic:dst_port = '3460'</td>
<td>network traffic to port '3460'</td>
</tr>
</tbody>
</table>

Note that (1) the nine observable objects are a subset of 102 objects described on the PIVY report [10] and Trend Micro [23]. (2) The empirical study uses lightweight command-line tools (CLT) to collect observable data. These tools include netstat, Windows Management Interface Command (wmic), PowerShell, Logparser, and Sysinternals Utilities. For demonstration, Figure 12 shows the observable evidence detected on the Windows registry, which indicates PIVY has created a registry key for autorun, which maintains the persistence of threats.

Figure 12. Observable evidence showing on Windows registry

C. Framework Deployment Diagram

The deployment diagram for detecting APTs is shown in Figure 13, which describes the system components after implementation. To facilitate the discus, we have added two teams in the diagram. A red team (red icon on the figure) is an independent group that challenges an organization to improve its effectiveness by assuming an adversarial role or point of view. The red team will: (1) Simulate APTs. Set up a controlled environment, e.g., using virtual machines, to simulate attacking scenarios, e.g., APT uses Poison Ivy. (2) Set up a Trusted Automated Exchange of Intelligence Information (TAXII) server [9]. It stores PIVY data models in a local repository [24]. (3) Serialize and de-serialize STIX JSON content using a TAXII client and Python APIs [25]. A blue team (black icon on the figure) is a group of individuals who perform an analysis of information systems to ensure the security of SUIs. Specifically, the blue team sets up a FED-FSM server to host FED-FSM execution frameworks, an observable data server to collect logs from an SUI and a TAXII/FED-FSM client [26] that executes the TADE and visualizes FED-FSM models. The empirical study adopts a Python framework for developing and running FSM-based workflows on AWS Lambda [27]. The framework provides a means to check a state machine's logic and monitor executions.

Figure 13. The deployment diagram for detecting APTs

VII. RELATED WORK

There have been many attempts to develop frameworks to systematically detect APTs. Bhatt et al. presented a framework [28] that models multi-stage attacks in a way that both describes the attack methods as well as the anticipated effects of attacks. The foundation to model behaviors is by the combination of the Intrusion Kill-Chain attack model and defense patterns. Haq et al. [29] described a computerized method in which one or more received objects are analyzed by an APT detection center to determine if the objects are APTs. The analysis may include the extraction of features describing and characterizing features of the received objects. The extracted features may be compared with features of known APT malware objects and known non-APT malware objects to determine a classification or probability of the received objects being APT malware. Wan et al. proposed a network gene-based framework [30] to describe the semantic-rich network behavior patterns of network applications. It took advantage of the latest advances in the fields of protocol reverse analysis, cloud computing, and big data processing, with automatic analysis and extraction of network genes, and data stream computing-based network gene real-time processing. Vert et al. [31] applied an advanced state machine engine to the analysis of state variables that can detect the presence of APTs and other malware. The Finite Angular State Velocity Machine (FAST-VM) can model and analyze large amounts of state information over a temporal space. The approach can analyze and model large amounts of data over time. Friedberg et al. applied a kind of black-list approach and only considered actions and behavior that match well-known attack patterns and signatures of malware traces [32]. They proposed an anomaly detection technique that keeps track of system events, their dependencies, and occurrences, and thus, the technique can learn the normal system behavior over time and report all actions that differ from the created system model.

None of the aforementioned APT detection frameworks are practical since they lack the essential characterizations of a framework for automation, including the scalability of
architecture [28][31][32], the formalization attacks of APT features [28][29][31], and the diversity of observable objects of SUI [30][32].

VIII. CONCLUSION

The paper presents a new formal approach that uses FED-FSM to detect APTs. The FED-FSM models are driven by real-world knowledge of adversary tactics and techniques stored in a shared repository. Instead of monitoring APTs directly, the approach infers the APT's status by analyzing the forensic evidence that malicious actors left on digital devices. The approach requires us systematically collecting crime activity logs, extracting evidence from logs, and formalizing digital forensic evidence. Two types of digital forensic evidence are defined in the paper, shared CFE objects, and observed CFE objects. These two objects are the drive force of FED-FSM. A demo program that is implemented in Java can be accessed at [33]. Note that the guided conditions of transitions in FED-FSM are predefined in FED-FSM using patterns. In future work, we are interested in investigating artificial intelligence-based approaches to discover patterns from shared CFE objects automatically and match patterns with observed CFE objects.

ACKNOWLEDGMENT

The work is supported in part by the National Science Foundation 1714261, 2039289, and the Office of Justice Programs 2019-DF-BX-K00.

REFERENCES


Minyang Yi*, Guisheng Fan*, Huiqun Yu†‡, Xingguang Yang†‡

*Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
†Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai 201112, China
‡Shanghai Engineering Research Center of Smart Energy, Shanghai, China

Abstract—Just-in-time software defect prediction (JIT-SDP) is an active research topic in the field of software engineering, aiming at identifying defect-inducing code changes. Most of the current JIT-SDP work focused on model construction. It is often ignored that the performance of classifiers often depends on high quality data. In this paper, we first investigate the impact of the class overlap problem on the performance of the classifiers in JIT-SDP, and propose a new effective preprocessing method (IKMCCA-TL) combining improved K-Means clustering cleaning approach and Tomek-link method. In order to objectively estimate the impact of class overlap on the classifiers in JIT-SDP, we conduct a large-scale empirical study on the data sets of six open source projects and compare the performance of LR, RF and KNN classifiers by using IKMCCA or KMCCA or NCL and without cleaning data. Experimental results show that after removing overlapping instances, the performance of the classifiers is significantly improved in terms of balance, recall and AUC and our proposed method achieves the best performance.

Index Terms—Just-in-time software defect prediction, class overlap, K-Means clustering, Tomek-link

I. INTRODUCTION

Software defect prediction technology is one of the most popular research topics among academic and industrial organizations [1]. In the development process of large-scale software, developers cannot avoid software defects. Defects in the software cause great harm and loss to users, customers and enterprises. In order to minimize the defects in the software system, developers will put a lot of effort into testing the software.

Just-in-time software defect prediction (JIT-SDP) is a special software defect prediction (SDP), which is at the software change level instead of the module level (for example, function, file, or class level) and it refers to the technology that predicts whether there are defects in each code change submitted by the developer [2]. Once the software change that caused the defect is implemented, it will be identified in JIT-SDP.

In SDP, researchers have done a lot of research on class imbalance and noise cleaning, and they have begun to study the problem of class overlap. Tang et al. [3] proposed a K-Means clustering cleanup method to clean noise instances by calculating the noise factor value \( NF_i \) of each instance for each cluster, and deleting the top p\%. Chen et al. [4] applied Neighbor Clean Learning (NCL) rules to remove overlapping instances for SDP. Gong et al. [5] proposed an improved K-Means clustering cleanup method (IKMCCA) to solve the problem of class overlap and class imbalance in SDP.

The main research work in JIT-SDP includes model construction and feature selection. In order to investigate the impact of the class overlap problem in JIT-SDP, we firstly uses the NCL, KMCCA and IKMCCA methods to process the data, and check whether the performance of the three classifiers is affected. Considering that overlap usually occurs near the decision boundary and removing important boundary instances will reduce the learning process, we propose an effective cleaning approach (IKMCCA-TL) combining IKMCCA method and Tomek-link pair method.

The rest of this article is organized as follows. Section II introduces related work. Section III introduces the proposed algorithm IKMCCA-TL. Experimental setup is described in section IV. Experimental results and discussion are presented in section V. Section VI describes the threats to validity. The conclusion and future work are presented in section VII.

II. RELATED WORK

A. Just-in-time Software Defect Prediction

The idea of JIT-SDP was proposed by Mockus et al. [6] and scores of change metrics to predict whether changes are defect-inducing or clean was designed by them. Kamei et al. [2] performed a large-scale empirical study in JIT-SDP. They collected eleven data sets which include six open-source projects and five commercial projects.

Subsequently, various methods were proposed to improve the performance of the prediction model for JIT-SDP. Yang et al. [7] validated the availability of progressive sampling in the JIT-SDP issue which can reduce the size of the defect data sets and reduce the cost of data sets acquisition. Chen et al. [8] proposed a JIT-SDP model MULTI based on multi-objective optimization algorithm NSGA-II to generate optimal solutions. The two goals of MULTI optimization are designed through the benefit-cost analysis. Yang et al. [9] proposed three optimal solutions selection strategies: benefit priority (BP), cost priority (CP), and a compromise between cost...
and benefit (CCB) to improve the performance of MULTI. Yang et al. [10] proposed a differential evolution (DE) based supervised method DEJIT to build JIT-SDP models which can significantly improve the effort-aware prediction performance in the three evaluation scenarios.

B. Class overlap

Class overlap means that some instances of different classes in the training data are close or even overlapped in the distribution space and often results in poor class boundaries and affects the performance of the learner. In the current SDP research, the problem of class overlap is mainly regarded as data quality or noise detection. Tang et al. [3] proposed a cluster-based noise detection method, which uses the outlier detection method to calculate the noise factor (NF), and removed the top \( p \% \) of NF. Chen et al. [4] proposed Neighborhood Clean Learning (NCL) to solve the problems of class overlap and class imbalance. The experimental results show that, compared with the existing learning methods, the new learning model can obtain the best values in terms of G-mean and AUC. In order to take into account the effects of class overlap and class imbalance, Gong et al. [5] proposed an improved K-Means clustering cleanup method (IKMCCA) to solve the problem of class overlap and class imbalance in SDP. Experiments have proved that compared to KMCCA and NCL methods, the IKMCCA method can obtain the best performance.

III. IKMCCA-TL

The evaluation model of IKMCCA-TL is shown in Fig 1. In the following section, we elaborate on the details of IKMCCA-TL.

Fig. 1. The evaluation model of IKMCCA-TL

A. IKMCCA

In order to consider both the impact of class overlap and class imbalance at the same time, Gong et al. [5] proposed an improved K-Means clustering cleanup method (IKMCCA) to solve the problem of class overlap and class imbalance in traditional defect prediction. In the step of deleting overlapping instances in the IKMCCA algorithm, if the percentage of defective instances in the cluster \( i \) is lower than \( p \% \), delete the defective instances in the cluster. On the contrary, instances without defects in this cluster will be deleted.

B. Tomek-link pair

The Tomek-link undersampling algorithm is used to eliminate boundary instances [11]. Given two instances \( t_i \in T \) and \( t_j \in T \) belonging to different classes, let distance \( (t_i, t_j) \) be the distance between them. If a pair of instances \( (t_i, t_j) \) does not exist distance \( (t_i, t_l) \) or distance \( (t_l, t_j) \) for any other instance \( t_l \in T \), then this pair of instances is called a Tomek-link pair.

C. IKMCCA-TL

The overlap often occurs near the decision boundary in the case of class overlap. In this case, the excessive elimination of boundary instances will drift the decision boundary between the minority class and the majority class, which in turn will reduce the learning process. Therefore, we improve the IKMCCA algorithm and combines the Tomek-link pair to make it possible to balance the data distribution, without distorting the decision boundary, and remove only unimportant boundary instances and unimportant redundant most instances. The pseudo code of IKMCCA-TL is shown as Algorithm 1.

IV. EXPERIMENTAL SETUP

In this paper, the problem of class overlap is studied in JIT-SDP, and the following two research questions are designed.

- RQ1: How does class overlap influence the prediction performance of the basic classifiers in JIT-SDP?
- RQ2: Why our proposed method (IKMCCA-TL) is more effective in reducing the impact of class overlap on the classifiers?

The experimental hardware environment is Intel(R) Core(TM) i7-10875 CPU@ 2.30GHz; RAM 16.00GB. The experimental code is written in Python.

A. Data Sets

The experiment considers the data sets of six open source projects shared by Kamei et al. [2], which have been widely used in JIT-SDP studies. The data set comes from 6 open source projects, which is shown in the Table I.

The data sets contains 14 change metrics. These characteristics are briefly introduced in [2], which involve five dimensions: diffusion, size, purpose, history, and experience.

B. Performance Indicators

In order to explore the influence of class overlap on the learner, we use three performance measures including Balance, Recall and AUC.
Algorithm 1: IKMCCA-TL

Input: training set: $T$, the parameter $m$.
// $m$ is used to determine the number of clusters
Output: a clean training set: $T_{\text{new}}$

begin
  $n =$ the number of instances in $T$
  $d =$ the number of defective instances in $T$
  $p = d/n$
  $k = \lfloor n/m \rfloor$
  using K-means algorithm to divide $T$ into $k$ clusters
  for $i = 1 \rightarrow k$
    find the instance pairs that are the Tomke-link pairs in cluster $i$
    compute the ratio $r$ of of defective instances to all instances in cluster $i$
    if $r > p$ then
      delete the non-defective instances of the Tomke-link pairs in cluster $i$
    else
      delete the defective instances of the Tomke-link pairs in cluster $i$
  end

$T_{\text{new}}$ is the set combining the remaining instances in each cluster
end

Table 1: The Information of Data Sets

<table>
<thead>
<tr>
<th>Project</th>
<th>Period</th>
<th>#defective changes</th>
<th>#changes</th>
<th>%defect rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUG</td>
<td>1998/08/26~2006/12/16</td>
<td>1696</td>
<td>4620</td>
<td>36.71%</td>
</tr>
<tr>
<td>COL</td>
<td>2002/11/25~2006/07/27</td>
<td>1361</td>
<td>4455</td>
<td>30.55%</td>
</tr>
<tr>
<td>JDT</td>
<td>2001/05/02~2007/12/31</td>
<td>5089</td>
<td>35386</td>
<td>14.38%</td>
</tr>
<tr>
<td>MOZ</td>
<td>2000/01/01~2006/12/17</td>
<td>5149</td>
<td>98275</td>
<td>5.24%</td>
</tr>
<tr>
<td>PLA</td>
<td>2001/05/02~2007/12/31</td>
<td>9452</td>
<td>64250</td>
<td>14.71%</td>
</tr>
<tr>
<td>POS</td>
<td>1996/07/09~2010/05/15</td>
<td>5119</td>
<td>20431</td>
<td>25.06%</td>
</tr>
</tbody>
</table>

C. Parameter Setting

In IKMCCA and IKMCCA-TL methods, we set the percentage $p\%$ to the ratio of defective instances, and the parameter $m$ is set as 20, which is the same as the references [5]. The parameter settings of KMCCA and NCL methods are the same as references [3] and [4]. 85% of the instances are randomly selected as training data, and rest instances are used as test data. For eliminating the randomness of the experiment, the experiment is repeated 20 times.

V. Experimental Results and Discussion

This section answers the questions raised in Section IV through experiments. And data processing is the same as [2].

A. Analysis for RQ1

In order to test the degree of impact of overlapping instances in software defect data sets on the performance of the classifiers, we adopt RF, KNN and LR classifiers.

As shown in the table II, through the experimental results, we can find that after using the NCL method to remove overlapping instances, the value of Recall increased by 2.9%, 4.6% and 5% respectively on LR, RF and KNN classifiers. The value of AUC increased by 1%, 1.8% and 2.2% respectively. The value of Bal increased by 1.9%, 3.1% and 3.4% respectively. KMCCA method which only remove the noise instances doesn’t achieve much improvement. The IKMCCA method which solves the problem of class overlap and class imbalance achieves better performance than the NCL method.

As shown in experimental result, the class overlap problem will have a serious impact on the performance of the classifier in JIT-SDP. When the overlapping instances of the class are removed, the performance of the classifier will be greatly improved. Considering the problem of class overlap and class imbalance at the same time, the classifier will perform better.

B. Analysis for RQ2

The existence of important boundary instances is also important for accurately defining the decision boundary. In the case of class overlap, the overlap often occurs near the decision boundary. In this case, the excessive elimination of boundary instances will drift the decision boundary between the minority class and the majority class, which in turn will reduce the learning process. In order to solve the above problems, we propose IKMCCA-TL method which only removes unimportant boundary instances and unimportant redundant instances.

Compared with the IKMCCA method, the Recall value of the IKMCCA-TL method is increased by 5.5%, 5.4% and 5.4% respectively on LR, RF and KNN classifiers. The value of AUC is increased by 0.7%, 0.8% and 0.6% respectively. The value of Bal is increased by 1.9%, 2.5% and 1.4% respectively.

As shown in experimental result, when we only delete unimportant overlap instances of the decision boundary instances, the performance of classifiers can be better.

VI. Threats to Validity

External validity. The results of the experiment can’t be guaranteed to apply to all other defect data sets. More data sets should be mined to verify the generalization of experimental results.

Construct validity. Three indicators, including Recall, AUC and bal, are used to reflect the performance of the classifier, which is also widely used in [5].

Internal validity. The threats to internal validity are mainly from experimental code. The mature python libraries are used and the code is checked to reduce the errors.

VII. Conclusions and Future Work

In JIT-SDP, the performance of the classifier often depends on high-quality data. In the past, the impact of overlapping classes on learning models was ignored. Therefore, we propose a method IKMCCA-TL that can better remove unimportant boundary instances and unimportant redundant instances, and
investigate whether NCL, KMC C, IKMCCA and IKMCCA-TL methods can improve the performance of the classifiers. We conduct a large-scale empirical study on data sets of six open source projects. Experimental results show that using these methods to eliminate overlapping instances can achieve significantly better performance in terms of bal, Recall, and AUC. And our proposed method IKMCCA-TL can better improve the performance of the classifiers by eliminating class overlapping instances.

In the future, more data sets from commercial projects will be mined to verify the generalization of experimental results.

ACKNOWLEDGMENT

This work was partially supported by the National Natural Science Foundation of China under Grant nos. 61772200, Shanghai Municipal Natural Science Foundation under Grant nos. 17ZR1406900 and 17ZR1429700.

REFERENCES


FSSRE: Fusing Semantic Feature and Syntactic Dependencies Feature for Threat Intelligence Relation Extraction

Xuren Wang, Mengbo Xiong*
Information Engineering College
Capital Normal University
Beijing, China
{wangxuren, 2181002015}@cnu.edu.cn

Famei He*
Library
Beijing Institute of Technology
Beijing, China
hefm@bit.edu.cn

Peian Yang*, Binghua Song, Jun Jiang, Zhengwei Jiang
Institute of Information Engineering
Chinese Academy of Sciences
School of Cyber Security
University of Chinese Academy of Sciences
Beijing, China
{yangpeian, songbinghua, jiangjun, jiangzhengwei}@iie.ac.cn

Zihan Xiong
Information Engineering College
Capital Normal University
Beijing, China
2181002064@cnu.edu.cn

Abstract—Threat intelligence relation extraction plays an important role in threat intelligence text analysis and processing. To extract the relation between two threat entities in a sentence, we develop a novel framework called FSSRE which fuses semantic feature and syntactic dependencies feature for threat intelligence relation extraction. We utilize graph convolutional networks (GCN) to extract syntactic dependencies features, and utilize Sentence-BERT to extract contextual semantic features. To keep vital information with irrelevant content removed to the most extent, we further apply a novel pruning strategy, SDP-VP, to the input tree. With retaining the shortest path and nodes that are $K$ hops away from nodes on the shortest path, we give the edge connected to the verb nodes a weight of $w$ times. We create an advanced persistent threat (APT) intelligence entities and intra-sentence relations dataset, APTER-SENT, for that there is no public dataset can be used for relation extraction research in the threat intelligence field. Experimental results on APTER-SENT demonstrate improved performance over competitive baselines. At the same time, we also conducted experiments on the SemEval-2010 dataset. The results of the experiment indicate that our method is still effective on this dataset.

Keywords—relation extraction, threat intelligence, APTER-SENT, GCN, Sentence-BERT, SDP-VP

I. INTRODUCTION

With the rapid development of network and information technology, new types of threats and attacks represented by Advanced Persistent Threat (APT) are showing a continuous and expanding development trend. APT attacks mainly use special Trojan to target computers to steal confidential information, commercial information of important enterprises, and destroy network infrastructure. Threat intelligence is a kind of evidence-based knowledge, including context, mechanism, labeling, meaning, and recommendations that can be implemented. Threat intelligence reports usually describe a malicious organization using some malicious software to launch an attack. That is, the report contains information such as attacker, target, purpose, and approach. The report also contains file HASH (e.g. 356A192B7913B04C54574D18C28D46E6 395428 AB), encryption algorithm (e.g. AES128-ECB), counter measure (keep CMS plugins up-to-date), etc. Security companies release massive amounts of threat intelligence every day. Most of threat intelligence is presented in text, which cannot visually show the connections between the attack events. It is not conducive to the rapid perception of abnormalities by security operators.

In order to help researchers quickly understand the connection between new threat events and previous threat events, it is very important to design algorithms that can extract the relationships between threat intelligence entities from a large number of documents. And the task of extracting threat intelligence relation is one of the key tasks in constructing a knowledge graph of threat intelligence. Although the effect of relation extraction in the general field is good, there are still some problems in relation extraction in the field of threat intelligence: (1) There is no public dataset that can be used for relation extraction research in the threat intelligence field; (2) The longer sentence length of threat intelligence text makes it difficult to fully and efficiently extract sentence features and not all tokens in the sentence are related to the relation between the entity pair of target. (As shown in TABLE I, the sentence length of threat intelligence text is longer than that of general domain text); (3) Because the threat intelligence text contains file HASH, encryption algorithm, counter measure and other professional domain information, the Out-Of-Vocabulary (OOV) problem will inevitably occur in the word embedding process. However, the existing relation extraction model cannot solve the above problems.

TABLE I. SENTENCE LENGTH STATISTICS FOR OUR DATASET APTER-SENT (THREAT INTELLIGENCE TEXT) AND THE SEMEVAL-2010 DATASET

<table>
<thead>
<tr>
<th>dataset</th>
<th>Average Sentence Length (Token)</th>
<th>Average Sentence Length (Char)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SemEval-2010</td>
<td>19</td>
<td>85</td>
</tr>
<tr>
<td>APTER-SENT</td>
<td>29</td>
<td>184</td>
</tr>
</tbody>
</table>

DOI reference number: 10.18293/SEKE2021-088
We regard the threat intelligence relation extraction task as a multi-classification task and construct a new model to extract relation. This model fuses semantic feature and syntactic dependencies feature for threat intelligence relation extraction. Graph convolutional networks (GCN) is used to capture the syntactic dependencies of the text. We observe that the longer sentence length of threat intelligence text makes it difficult to extract sentence features sufficiently and effectively, and not all tokens in the sentence are related to the relation between the entity pairs of target. In order to solve this problem, we need a pruning method to incorporate relevant information with maximally removing irrelevant content. We also found that the verbs in the text often play a decisive role in the classification of the relation between entities, so we propose a new pruning method of verb-based shortest dependent path pruning (SDP-VP). SDP-VP keeps the shortest dependent path and its K-hop nodes, and gives the edge connected to the verb nodes a weight of \( w \) times to form a new dependency graph, which help extract relation between entities more efficiently. But at the same time, there are new problems. If we only use structural information to extract the relations between entities in the longer text, we will lose semantic information, but semantic information is also very important for the task of relation extraction. Therefore, we use Sentence-BERT [1] to capture contextual semantic information, and at the same time alleviate the Out-Of-Vocabulary (OOV) problem caused by special text like HASH value, encryption algorithm and counter measure in the field of threat intelligence. Finally, semantic feature and syntactic dependencies feature are fused to classify the relation.

II. RELATED WORK

Existing general domain relation extraction methods can be divided into two categories, sequence-based and dependency-based.

Sequence-based methods mainly rely on word sequences. This method uses the entire text as input, and uses convolutional neural networks (CNN), long and short-term memory networks (LSTM), etc. to capture the con-textual information of the text, and then extract the relation. Zhang et al. [2] first combines the LSTM sequence model with an entity position attention mechanism that is more suitable for relation extraction. Verga et al. [3] used Transformer to encode the context, and each encoded word generated two position-specific representations through the head and tail multi-layer perceptron (MLP) to achieve relation extraction. The above methods all promote the progress of relation extraction, but the above models only uses the contextual semantic information in the text and cannot capture the dependency information of the sentence.

The dependency-based model incorporates the dependency tree into the model. For example, Song et al. [4] proposed a graph-state LSTM model, which uses parallel state to model each word, and enriches state values through messaging to achieve entity extraction. Zhang et al. [5] proposed an extended graph convolutional network. In order to hold relevant information and remove irrelevant content to the maximum, the model further applies a new pruning strategy to the input tree. The pruning method is to retain the K-hops nodes that are the shortest dependent path between two entities. This pruning strategy can be visually called "hard pruning". Its advantage is to reduce the nodes with less information, reducing the size of the dependent graph, and improve the efficiency of model processing. Guo et al. [6] proposed an Attention Guided Graph Convolutional Networks (AGGCNs) model that directly takes completely dependent trees as input. However, a "soft pruning" method is proposed, which automatically learns how to selectively focus on the relevant substructures that are useful for the relation extraction task. The advantage of this method is that it does not sub-tract any node on the dependency graph, but assign different weight values to the edges on the graph, so that the model extracts text representations that are more beneficial to the relation extraction task. Can [7] proposed a novel model, which is based on the basic information in the SDP enhanced with information selected by several attention mechanisms with kernel filters, namely RhSP (Richer-but-Smarter SDP). But the extracted context information and the extracted dependency structure information are both extracted based on the input pruned dependency graph, and the original context information of the text is not obtained. In essence, Can's work is a dependency-based method. Context semantic information is very important for relation extraction, which is ignored in the above methods. Moreover, the above papers all propose relation extraction models for general domain datasets, which are directly used to extract the relationship between entities in threat intelligence domain. For example, threat intelligence sentences are usually longer than general domain sentences, and it is not easy to extract sentence features for relation extraction. It is necessary to propose a suitable pruning method to solve this problem, and pruning method can also improve the efficiency of model processing.

Threat intelligence entity relation extraction is relations’ extraction for specific fields. [8] proposed a system for creating semantic triples on cyber security text, using deep learning methods to extract possible relation. Du et al. [9] propose a knowledge graph for People-Readable Threat Intelligence recommendation (PRTIRG) and incorporates knowledge graph representation into PRTI recommender system for click-through prediction. Wang et al. [10] propose a distant supervision relationship extraction method RL-ET-PCNN-ATT based on the PCNN-ATT model. Verbs are very important for judging the relationship between threat intelligence entities, which is not discussed in the above methods.

III. DATASET

In order to solve the problem that there is no public dataset that can be used for relation extraction research in the threat intelligence field, we constructed threat intelligence entities and intra-sentence relations dataset, APTER-SENT, through manual annotation. APTER-SENT includes annotations for cybersecurity-related entities and relations. The APT reports we marked come from web. They have been issued by major security vendors (such as Kaspersky, FireEye, etc.) since 2008. We have marked 12,906 sentences in reports published between 2015 and 2020. After preprocessing, 23,225 triples are obtained. To ensure the quality of the dataset, a three-person cross-check is used in manual annotation. We did not use automatic and semi-automatic annotation solutions, although they are better at facing texts in general domains, they will generate a lot of noise data when facing texts in the threat intelligence domain. According to STIX2.0, "National Standard of the People's
Republic of China—Information Security Technology - Cybersecurity Threat Information Format Specification” and domain experts' knowledge, we predefined 36 entity categories. And we predefined 8 relation types. TABLE II shows 36 predefined entity types and their corresponding examples and quantities. And we predefined 8 relation types as shown in TABLE III. The example is only one case in the instance relationship but not all, such as "located" which can also indicate that the file is located in the operating system such as "The "d3d9.dll" file is malicious and is loaded into memory of Windows".

<table>
<thead>
<tr>
<th>Entity type</th>
<th>Example</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>threatActor_name</td>
<td>APT10</td>
<td>1447</td>
</tr>
<tr>
<td>threatActor_aliases</td>
<td>Menupass</td>
<td>62</td>
</tr>
<tr>
<td>program_language</td>
<td>JavaScript</td>
<td>81</td>
</tr>
<tr>
<td>security_team</td>
<td>AntiyCERT</td>
<td>349</td>
</tr>
<tr>
<td>vulnerability_cve</td>
<td>CVE-2014-0160</td>
<td>43</td>
</tr>
<tr>
<td>vul_aliases</td>
<td>Heartbleed</td>
<td>38</td>
</tr>
<tr>
<td>encryption_algo</td>
<td>AES128-ECB</td>
<td>157</td>
</tr>
<tr>
<td>sample_name</td>
<td>control.exe</td>
<td>711</td>
</tr>
<tr>
<td>sample_function</td>
<td>Delete files</td>
<td>513</td>
</tr>
<tr>
<td>government</td>
<td>FBI</td>
<td>103</td>
</tr>
<tr>
<td>target_crowd</td>
<td>military</td>
<td>181</td>
</tr>
<tr>
<td>attack_activity</td>
<td>Watering hole</td>
<td>270</td>
</tr>
<tr>
<td>counter_measure</td>
<td>Keep CMS plugins up-to-date</td>
<td>118</td>
</tr>
<tr>
<td>sub_activity</td>
<td>Encrypts this data</td>
<td>771</td>
</tr>
<tr>
<td>OS_name</td>
<td>Windows</td>
<td>149</td>
</tr>
<tr>
<td>email_evil</td>
<td><a href="mailto:acc.signedin.send@gmail.com">acc.signedin.send@gmail.com</a></td>
<td>31</td>
</tr>
<tr>
<td>malware</td>
<td>Reaver</td>
<td>1439</td>
</tr>
<tr>
<td>string</td>
<td>VIEWS0018x</td>
<td>342</td>
</tr>
<tr>
<td>domain_evil</td>
<td><a href="http://www.tashdqdxp.com">www.tashdqdxp.com</a></td>
<td>331</td>
</tr>
<tr>
<td>domain</td>
<td><a href="https://www.proofpoint.com">https://www.proofpoint.com</a></td>
<td>18</td>
</tr>
<tr>
<td>attack_goal</td>
<td>Steal e-mail and contacts</td>
<td>230</td>
</tr>
<tr>
<td>location</td>
<td>Western Europe</td>
<td>543</td>
</tr>
<tr>
<td>industry</td>
<td>Maritime industries</td>
<td>111</td>
</tr>
<tr>
<td>company</td>
<td>Microsoft</td>
<td>170</td>
</tr>
<tr>
<td>function</td>
<td>PostDown()</td>
<td>201</td>
</tr>
<tr>
<td>protocol</td>
<td>HTTP</td>
<td>170</td>
</tr>
<tr>
<td>person</td>
<td>Tom Smith</td>
<td>445</td>
</tr>
<tr>
<td>IP</td>
<td>192.168.1.206</td>
<td>6</td>
</tr>
<tr>
<td>IP_evil</td>
<td>98.126.156.210</td>
<td>71</td>
</tr>
<tr>
<td>md5</td>
<td>292843976600e8ad2130224d70356bfc</td>
<td>167</td>
</tr>
<tr>
<td>sha1</td>
<td>356a192b7913b04c5547d18c2884d6e6395428ab</td>
<td>16</td>
</tr>
<tr>
<td>sha2</td>
<td>12dedcdd853da9846014186e6b45d6a82badfc7fa4cbe444a010f682b5d</td>
<td>421</td>
</tr>
</tbody>
</table>

Numerous examples are utilized in Table III. The example is only one case in the instance relationship but not all, such as "located" which can also indicate that the file is located in the operating system such as "The "d3d9.dll" file is malicious and is loaded into memory of Windows".

<table>
<thead>
<tr>
<th>Relation type</th>
<th>Example</th>
<th>Counts</th>
</tr>
</thead>
<tbody>
<tr>
<td>attack</td>
<td>The Gaza cybergang’s attacks have never slowed down and its typical targets include government entities/embassies, oil and gas.</td>
<td>652</td>
</tr>
<tr>
<td>located</td>
<td>The energy sector in Europe and North America is being targeted by a new wave of cyber attacks.</td>
<td>555</td>
</tr>
<tr>
<td>part_of</td>
<td>Costin Raiu, director of Global Research and Analysis Team at Kaspersky Lab, was the first to find a code connection between APT17 and the backdoor in the infected.</td>
<td>2290</td>
</tr>
<tr>
<td>occur_time</td>
<td>WhiteBear focused on various embassies and diplomatic entities around the world in early 2016.</td>
<td>501</td>
</tr>
<tr>
<td>use</td>
<td>Gaza cybergang started using the CVE 2017-0199 vulnerability which enables direct code execution.</td>
<td>1638</td>
</tr>
<tr>
<td>launch</td>
<td>Proofpoint detected and blocked spearphishing emails from Leviathan targeting a US shipbuilding company</td>
<td>657</td>
</tr>
<tr>
<td>goal</td>
<td>Gaza cybergang started using the CVE 2017-0199 vulnerability which enables direct code execution.</td>
<td>1360</td>
</tr>
<tr>
<td>find</td>
<td>CVE-2017-8759 is the second zero-day vulnerability used to distribute FINSPY uncovered by FireEye in 2017.</td>
<td>459</td>
</tr>
</tbody>
</table>

APTER-SENT and SemEval-2010 dataset are both focused on annotating intra-sentence relations. The difference is that APER-SENT is a threat intelligence field dataset and SemEval-2010 is a general field dataset. The former contains at least one relation in each sentence, while the latter contains only one relation (1.8 vs. 1.0 relations per sentence). TABLE IV gives a comparison of statistics among the two datasets. SemEval-2010 dataset only marked entity position but did not define the entity type. For the convenience of comparison, we counted the number of entities, the entities counted contain the entity types in Table II. The statistics of the number of relations in the two datasets includes the "Other" type.

<table>
<thead>
<tr>
<th>Statistics</th>
<th>APER-SENT</th>
<th>SemEval-2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entities</td>
<td>12873</td>
<td>21434</td>
</tr>
<tr>
<td>Relations</td>
<td>23225</td>
<td>10717</td>
</tr>
<tr>
<td>Sentences</td>
<td>12906</td>
<td>10717</td>
</tr>
<tr>
<td>Relations/Sentence</td>
<td>1.8</td>
<td>1.0</td>
</tr>
</tbody>
</table>
IV. MODELS

The overall architecture of the proposed FSSRE is illustrated in Fig. 1. It is mainly composed of four parts: (1) Word Processor which extracts word-level semantic features; (2) Regional Dependency Feature Extractor that learn syntactic dependencies information; (3) Semantic Feature Extractor that learn contextual Semantic information; (4) Relation Classifier that classifies relation between entity pairs into predefined categories.

A. Word Processor

In order to fully capture the threat intelligence text features, we use four features to form the final word embedding:

a. We use the pre-trained 300-dimensional GloVe vectors.
b. We use the BERT [11] model to embed words in the text to alleviate the OOV problem.
c. Part-of-speech (POS) being very important for the relation extraction task.
d. Entity tags are beneficial for relation extraction between entities.

Fig. 1. Overview of FSSRE

As shown in Fig. 1, the input is a sequence of sentences \( S = \{t_1, t_2, \ldots, t_n\} \), where \( t_n \) represents the n-th token in the sentence. Our final input representation for token \( t_n \) is:

\[
t_n = G_n + B_n + P_n + E_n
\]  

(1)

Where \( G_n \) is the GloVe token embedding vector, \( B_n \) is the BERT embedding vector, \( P_n \) is the POS embedding and \( E_n \) is the NER embedding. Some modules in Word Processor can be replaced with a wide range of different neural network designs. For example, FastText [12] can be used to replace GloVe. It is very flexible and can be adjusted according to needs.

B. Regional Dependency Feature Extractor

a) SDP-VP: To capture the regional dependency feature of threat intelligence text, we use graph convolutional networks (GCN) to process the dependency graph generated by the text. We have noticed that tokens of the verb part are crucial in determining the relation between entities, which plays a decisive role in judging the relation between entities. We propose a new pruning method SDP-VP. SDP-VP keeps the shortest dependent path and the nodes that are \( K \) hops away from the node above it, and gives the edge of the verb part-of-speech node a weight of \( w \) times to form a new dependency graph. As shown in the Fig. 2, assuming that nodes \( N_1, N_5 \) and \( N_7 \) are verb nodes, the red path represents the shortest dependent path between entity \( E_1 \) and entity \( E_2 \). After SDP-VP pruning, in addition to the nodes on the shortest path will be retained, nodes that are \( K(K = 1) \) hops away from the shortest path will also be retained. And the weight of the edge with the verb part of speech node is \( w(w = 2) \) times the weight of other edges. Finally, the pruning result as shown in the Fig. 2 is obtained.

This method holds the advantages of "hard pruning" and "soft pruning", which can not only reduce the size of the dependent graph, improve the efficiency of model processing, but also give higher weight to the edges of the verb part of speech, making the model more concerned information that is more favorable to relation extraction. Following Zhang et al [5], we further merge the input sentence sequence \( S \) through the bidirectional long short-term memory network (BiLSTM) to obtain the word embedding with context information to obtain a new sentence sequence \( S_t = \{t_1, t_2, \ldots, t_n\} \). This BiLSTM contextualization layer is trained jointly with the rest of the network.

Fig. 2. The Processing of SDP-VP (\( K = 1, w = 2 \))

\[
\vec{h}_n = LSTM_{fw}(t_n, \vec{h}_{n-1})
\]  

(2)

\[
\vec{h}_n = LSTM_{bw}(t_n, \vec{h}_{n-1})
\]  

(3)

\[
l_n = [\vec{h}_n, \vec{h}_{n}]
\]  

(4)

For each word \( t_n \) of the sentence, the hidden layer state \( \vec{h}_n \) is obtained through the forward LSTM, and the hidden state \( \vec{h}_n \) is obtained through the backward LSTM, and the two are connected to obtain the hidden layer states \( l_n \). \( l_n \) is the word embedding to indicate the state of the hidden layer obtained after \( t_n \) passing BiLSTM.
The sentence sequence $S_L$ is pruned by SDP-VP to obtain a new sequence $P = \{p_1, p_2, \ldots, p_m\}$, $p_m$ represents the $m$-th token in the sequence after pruning.

b) Extract Dependency Graph Regional Dependency Feature: In order to explicitly use structural information to further improve the model, we propose to use Stanford dependency parser creating a dependency tree for the input sentence. We use the dependency tree as the input sentence’s adjacency matrix and use GCN extracting regional dependency features. Graph convolutional network (GCN) was proposed by Kipf and Welling [13] in 2017. A GCN layer retrieves new node features by considering neighboring nodes’ features with the following equation:

$$h_v^{l+1} = ReLU\left( \sum_{v \in N(v)} (w^l h_v^l + b^l) \right)$$

(5)

Where $v$ is the target node and $N(v)$ represents the neighborhood of $v$, including $v$ itself; $h_v^l$ denotes the hidden feature of node $v$ at layer $l$; $W$ and $b$ are learnable weights, mapping the feature of a node onto adjacent nodes in the graph.

After applying an L-layer GCN over word vectors, we obtain hidden representations of each token $S_g = \{g_1, g_2, \ldots, g_m\}$. To make use of these word representations for relation extraction, following Zhang et al [5], we first obtain a sentence representation as follows:

$$G_{sentence} = f(S_g)$$

(6)

Where $f$ is a max pooling function that maps from $n$ output vectors to the sentence vector, and $G_{sentence}$ is the sentence representation through the GCN network. Similarly, we can obtain the entity representations. For the head entity and the tail entity, their representation $E_{head} \cdot E_{tail}$ can be computed as:

$$E_x = f(g_x)$$

(7)

Where $x$ is “head” or “tail”, $g_x$ represents the head entity representation (tail entity representation) after passing through the GCN.

We obtain the regional dependency feature representation by concatenating the sentence and the entity representations, and feeding them through a Multi-layer Perceptron (MLP):

$$h_{dependency} = MLP([G_{sentence}; E_{head}; E_{tail}])$$

(8)

C. Semantic Features Extractor

In order to capture the Semantic Features of threat intelligence text, we use Sentence-BERT [1] to process the input threat intelligence text to obtain the semantic information of the text. At the same time, Sentence-BERT is used to alleviating one of the key obstacles, Out-Of-Vocabulary (OOV) problem, to processing threat intelligence texts. Sentence-BERT (SBERT), a modification of the pre-trained BERT network that use siamese and triplet network structures to deriving semantically meaningful sentence embeddings that can be compared using cosine-similarity. Sentence-BERT can be flexibly replaced with other language model such as RoBERTa [14].

$$B_{sentence} = SBERT(S)$$

(9)

We obtain the semantic features representation by feeding $B_{sentence}$ through a Multi-layer Perceptron (MLP):

$$h_{semantic} = MLP(B_{sentence})$$

(10)

D. Relation Classifier

After obtaining dependent feature representation and semantic feature representation, we obtain the syntactic-semantic representation used for classification by concatenating them. This syntactic-semantic representation is then fed into a linear layer followed by a softmax operation to obtain a probability distribution over relations.

V. EXPERIMENTS

In this section, we present the experimental results of the proposed FSSRE. We first describe the experimental setup, the baselines we compare with, and experimental results. Finally, we also do experiments on the SemEval-2010 Task 8 dataset.

A. Experimental Setup

The BERT pre-training model we use is 768-dimensional BERT-Base-Cased. We concatenate together the last 4 hidden layers and sum them to get the final BERT word vector. We use the pre-trained 300-dimensional GloVe vectors to initialize word embeddings, and use embedding size of 30 for all other embeddings (i.e., POS, NER). We use the dependency parse trees, POS and NER sequences in our dataset (APTER-SENT). NER sequences are generated by manual annotation. POS sequences and the dependency parse trees were generated with Stanford Core NLP. We use 2 GCN layers and employ the ReLU function for all nonlinearities in the GCN layers and the standard max pooling operations in all pooling layers. In GCN model, we use the hidden layer of 200 as the output feed-forward layer. We set LSTM hidden size to 200. The APTER-SENT dataset contains many triples with the relation ”Other“, resulting in data imbalance. In order to mitigate the impact of data imbalance on the experimental results, we use the Undersampling method dividing the positive instances (the relation is pre-relation) and negative instances (the triple with the relation ”Other“) of the data. The ratio is adjusted to 1:1. We use cross-entropy as the loss function in Eq. (11). In the experiment, we tried the following four optimization functions: sgd, adagrad, adam, and adamax, which were verified by experiments: the adagrad optimization function was selected in the final model to achieve the best results.

$$f_{loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log \hat{y}_{ij}$$

(11)

Where $N$ represents the number of samples in a batch, $M$ represents the number of relationship categories, $y_{ij}$ represents the predicted value, $\hat{y}_{ij}$ represents the true value.
B. Results on APTER-SENT dataset

We compared FSSRE with two baselines:

- C-GCN [5]: A new pruning method keeps the nodes which are \( K \) hops away from the shortest path on the basis of the shortest dependent path. This model use LSTM+GCN network to extract text features for relation extraction.

- AGGCN [6]: This paper proposes a soft pruning method to automatically learn and selectively focus on related substructures for relation extraction.

As shown in Table V, for model C-GCN, We experimented with \( K \in \{-1,0,1,2,\cdots,10\} \) on our dataset (APTER-SENT), where when the number of pruning \( K = 2 \), the F1 value is the largest. For the model AGGCN, We experimented on our dataset (APTER-SENT). FSSRE surpasses the AGGCN model by 1.585\% (F1). Our model is an improvement based on the model C-GCN and surpasses the C-GCN model by 2.274\%. For a fair comparison with the compared model, we removed the BERT word embedding from our model, namely FSSRE (-BERT), and the experimental results are still better than the compared model, e.g., surpassing the AGGCN model by 1.139\% (F1). Although our model (FSSRE) precision is 0.25\% lower than AGGCN, but recall increased by 3.525\% and F1 increased by 1.585\%.

![Table V. Results on the APTER-SENT dataset](image)

We examine the contributions of three components: BERT, SDP-VP and Sentence-BERT. We removed Sentence-BERT, SDP-VP and BERT from FSSRE model successively and cumulatively for ablation experiment. Table VI shows the results. We can observe that adding either SDP-VP or Sentence-BERT improves the performance of the model. This suggests that both layers can assist FSSRE to learn better information aggregations, where the SDP-VP seems to be playing a more significant role. And we find that: (1) The Sentence-BERT contribute F1 with 0.233\%, (2) The SDP-VP layers improve F1 with 0.723\%. (3) The BERT layers improve F1 with 1.318\%.

![Table VI. An ablation study for FSSRE model on the APTER-SENT dataset](image)

In order to show the effect of pruning number \( K \) and weight \( w \) on the experimental results in our pruning method, we experimented with \( K \in \{-1,0,1,2,\cdots,10\} \) on our dataset (APTER-SENT), where \( K = -1 \) means input full tree. The experimental results are shown in Fig. 3. And we experimented with \( w \in \{1.0,1.5,2.0,2.5,\cdots,5.5,6.0\} \) on APTER-SENT, the experimental results are shown in Fig. 4. It can be seen from Fig. 3 that when the pruning number \( K = 1 \), the F1 value is the largest. It can be concluded from Fig. 4 that the F1 value is the largest when the weight \( w = 2.0 \).

![Fig. 3. Influence of pruning number \( K \) on experimental results (\( w = 2 \))](image)

![Fig. 4. Influence of weight \( w \) on experiment results (\( K = 1 \))](image)

C. Results on the SemEval-2010 Task 8 dataset

We also conducted experiments on the general domain public dataset SemEval-2010 Task 8. We use Stanford CoreNLP to preprocess the SemEval-2010 Task 8 dataset to generate dependency parse trees, POS and NER annotations. The other experimental settings are consistent with APTER-SENT. The experimental results are shown in Table VII. For a fair comparison with the baseline model, we removed the BERT word embedding from our model, namely FSSRE (-BERT) which still surpasses the AGGCN model by 1.5\% (F1). Compared with AGGCN, our model (FSSRE) improves by 2.2\% (F1). Experimental results prove that our method is still effective on this dataset.

![Table VII. Results on the SemEval-2010 Task 8 dataset](image)
For the SemEval-2010 Task 8 dataset, we also examine the contributions of three components: BERT, SDP-VP and Sentence-BERT. Similarly, we removed Sentence-BERT, SDP-VP and BERT from FSSRE model successively and cumulatively for ablation experiment. Table VIII shows the results: (1) The Sentence-BERT contribute F1 with 0.221%. (2) The SDP-VP layers contribute F1 with 0.554%. (3) The BERT layers contribute F1 with 1.990%.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
<th>F1 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSSRE(ours)</td>
<td>87.957</td>
<td>87.811</td>
<td>87.884</td>
</tr>
<tr>
<td>- Sentence-BERT</td>
<td>86.393</td>
<td>88.972</td>
<td>87.663</td>
</tr>
<tr>
<td>- SDP-VP</td>
<td>85.475</td>
<td>88.806</td>
<td>87.109</td>
</tr>
<tr>
<td>- BERT</td>
<td>83.952</td>
<td>86.318</td>
<td>85.119</td>
</tr>
</tbody>
</table>

In order to show the influence of the pruning number $K$ and weight $w$ on the experimental results in our pruning method. The experimental results are shown in Fig. 3. From the Fig. 3, it can be concluded that when the pruning number $K=1$, the F1 value is the largest. And we experimented with $w \in \{1, 0.1, 0.5, 2, 0.2, 5, \ldots, 5.5, 6, 0\}$ on the SemEval-2010 Task 8 dataset, the experimental results are shown in Fig. 4, from which we can draw a conclusion: when the weight $w = 2.0$, the F1 value is the largest.

VI. CONCLUSIONS

In this paper, we propose a new dataset APTER-SENT, which contains annotations for cyber security-related entities and intra-sentence relations. We have presented FSSRE, a novel model of relation extraction between two entities in a sentence that can simultaneously contain contextual semantic features and syntactically dependent features. We use GCN to extract syntactically dependent features and use Sentence-BERT to extract contextual semantic features, and then fuse the two to extract relation. In order to obtain the most useful information, we propose a new pruning method SDP-VP. We evaluated our model on APTER-SENT and SemEval-2010 task 8 dataset, then compared the results with very recent state-of-the-art models. The results demonstrated the advantage and robustness of our model. We aim to improve this model so that it can extract inter-sentence relations in future works.

ACKNOWLEDGMENT

We are grateful to the anonymous reviewers for their valuable comments and suggestions to significantly improve the quality of this paper. This work is supported by the National Key Research and Development Program of China (Grant No. 2018YFC0824801).

REFERENCES


AAMR: Automated Anomalous Microservice Ranking in Cloud-Native Environment

Zekun Zhang, Bing Li, Jian Wang, Yongqiang Liu
School of Computer Science,
Wuhan University,
Wuhan, China
e-mail: jianwang@whu.edu.cn

Abstract—Recently, it has become a trend for developers to build applications using the microservice architecture. The functionality of each application is divided into multiple independent microservices, which are interconnected to others. With the emergence of cloud-native technologies, such as Docker and Kubernetes, developers can achieve a consistent and scalable delivery for complex software applications. However, it is challenging to diagnose performance issues in microservices due to the complex runtime environments and the numerous metrics. In this paper, we propose a novel root cause analysis approach named AAMR. AAMR first constructs a service dependency graph based on real-time metrics. Next, it updates the anomaly weight of each microservice automatically. Finally, a PageRank-based random walk is applied for further ranking root causes, i.e., ranking potential problematic services. Experiments conducted on Kubernetes clusters show that the proposed approach achieves a good analysis result, which outperforms several state-of-the-art methods.

Keywords—Microservice, Anomaly detection, Root cause analysis, Cloud-native system

I. INTRODUCTION

Nowadays, microservice architectures (MSA) have become increasingly popular in large-scale software development following different computing paradigms like cloud computing, mobile computing, and edge computing. MSA-based software applications are decomposed into light-weighted, interconnected, independently deployed, and scalability-enabled microservices [1]. With the decomposition, the process of testing, deploying, and releasing becomes faster. However, as user requirements change, software code commits, and version updates become increasingly frequent. Many unexpected issues may arise, which have a significant impact on service quality and user experience. It is important for developers to figure out the root causes of system failures and mitigate them.

Traditionally, system failures are usually pinpointed by checking the log and event tracking, and then the performance issues are analyzed based on monitoring tools [2]. With the increasing scale and complexity of software, service dependencies also become increasingly complex, making these tools hard to achieve the needs of troubleshooting and diagnosis. In general, when an anomaly occurs in microservice systems, the anomaly detected is merely a symptom, and the root cause often hides from a larger underlying issue. Particularly, if a microservice becomes abnormal, e.g., response time delay or interruption of work, most of the microservices collaborated with it will be implicated. Therefore, it is necessary to detect undesirable performance problems and pinpoint the underlying anomalous microservice (root cause).

At present, the challenges of locating potential root causes are (i) Large volume of metrics: Communications between services are plenty and frequent, which cause a large volume of monitoring metrics (e.g., OpenStack exposes 17,608 metrics [3]). It is challenging to pinpoint the bottleneck from numerous and diverse metrics. (ii) Different failure sources: The failures might be caused by upstream or downstream tasks in the propagation direction. Besides, the wrong deployments and insufficient resource utilization can also cause failures. (iii) Highly dynamic in runtime: Due to the flexibility of microservices, the IP address of a microservice may dynamically change in creating a replica. The scalability of replicas further enlarges the service correlation and the complexity of locating anomalies.

Many existing works on root cause analysis have been reported. Most of these works [4-8] localize the root cause by constructing a service dependency graph (SDG) [10] based on monitored metrics. With the SDG, the anomalous microservices are commonly ranked by the similarity between back-end services and front-end services. However, services that have little impact on front-end services are missing in the diagnosis. As for metrics, parts of these works [5, 6] only use application-level metrics, which is insufficient for analysis. Some works [7, 8] consider multiple metrics while missing the key metrics ranking. To address these limitations, we propose a novel approach to detect anomalies and locate the root cause in microservice systems.

If there is an anomalous node in the service network, the nodes associated with it are likely affected. Inspired by the mRank [9] algorithm, we use adjacent nodes to represent the anomaly score of the target node. As for input, we collect multiple metrics, including system utilization and application-level metrics. Our goal is to localize the root cause and highlight the key anomalous metric, which helps developers diagnose system failures. We evaluate our approach on Kubernetes clusters and inject several common failures that occur in cloud-native systems. The results show that our approach outperforms several state-of-the-art methods in localizing accuracy. In summary, our contributions include:

• We extend the mRank algorithm for root cause analysis in microservices. Our method can automatically update the anomaly weights in SDG.

DOI reference number: 10.18293/SEKE2021-091
We evaluate our method in a cloud-native environment. The experimental results show that our approach has higher accuracy and faster than other baseline methods on the benchmark.

The remainder of this paper is organized as follows. Related works are summarized in Section II. Section III formulates the problem. We elaborate on our proposed approach in Section IV. Experiments and evaluations are included in Section V. The conclusion and future work are given in Section VI.

II. RELATED WORK

Root cause analysis for distributed systems has been devoted in the industry and academia for years. Existing approaches in this area can be approximately classified into four types.

Trace-based methods. Many tools and systems on end-to-end tracing like Dapper [11], Pinpoint [12], and EagleEye [13] collect the trace information. These tools can accurately record the execution path of programs and then locate the failure by detecting the source code or binary code. However, a large-scale system is usually developed by many teams with different languages over the years, and the overhead of modifying its source code is often too high [14].

Log-based methods. The system log is an important clue for analysis [2]. By parsing patterns and extracting features from event logs, Xu et al. [15, 16] built anomaly detection and identification models from historical data and used these models to analyze root causes. However, as the application flexibility increases, these methods are less effective in analyzing the anomalies in real-time.

Machine learning-based methods. Some researchers use the metrics collected as training data, instead of logs, to train models. Brândon et al. [17] constructed fault patterns from several fault injection methods. The anomalies are classified by comparing the similarity between the anomaly graph and fault patterns. Moreover, Du et al. [18] collected real-time performance data such as CPU, memory, response time, and package loss to build a model for anomaly detection. GRANO [19] created an anomaly analysis model and visualized the analysis result. But these approaches require collecting a large amount of data for model training, and these models cannot cover all anomalous patterns.

Graph-based methods. Many graph-based approaches are also proposed based on real-time performance metrics. For example, CloudRanger [6] constructed an impact graph based on the dynamic causal relationship. Microscope [5] added anomalous nodes into a candidate group and then ranked the anomalous nodes in the candidate group based on the correlation coefficients between nodes. But only application-level metrics are included in their works, which is insufficient for analysis. To solve such problems, MicroCause [20] used multi-metric and captured the sequential relationship of time series data, and MS-Rank [7] updated the weights of different metrics dynamically. These methods used forward, self, and backward random walk to heuristically locate root causes. Besides, Weng et al. [21] found that anomalies occur on both the service and physical level. MicroRCA [8] correlated anomalous performance symptoms with relevant resource utilization to represent service anomalies. However, MicroRCA cannot update the anomaly detection confidence (i.e., weights in SDG) automatically.

Similar to graph-based approaches, we also use a graph model and rank the anomalies using a random walk algorithm. In our approach, we automatically update the anomaly weights in SDG and output a two-phase ranking list that contains the anomalous nodes and metrics.

III. PROBLEM DEFINITION

To generalize the problem, we treat the microservice system as a “black box” that requires no domain knowledge, and the root cause analysis process is running independently. Many reasons can cause abnormal events in microservices, such as sudden increases in throughput, errors in code logic, and insufficient allocation of host resources. We refer to the process of diagnosing those anomalous nodes and the metrics responsible for the abnormal events as root cause analysis. The identification of anomalous nodes is regarded as root cause localization. We monitor the metrics change of all microservices in the system by default. These metrics are collected as a matrix in time window $T$. We denote the matrix as $M_i$ and $M_j$ stands for the metrics in column $k$. Our objective is to identify a set of root causes $V_c$ and rank the associated metrics for each root cause. The notations used in the paper are listed in Table I.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(V, E, W)$</td>
<td>Service dependency graph with weight matrix $W$</td>
</tr>
<tr>
<td>$M_i, M_j$</td>
<td>Metrics collected in $T$ and metrics in column $k$</td>
</tr>
<tr>
<td>$V_i, h_i$</td>
<td>Microservice node $i$ and the host node of $V_i$</td>
</tr>
<tr>
<td>$P, p_{ij}$</td>
<td>$P_{ij} = p_{ij}$, transition probability from $V_i$ to $V_j$</td>
</tr>
<tr>
<td>$RT_i$</td>
<td>Response time series of $V_i$ in $T$</td>
</tr>
<tr>
<td>$\Delta, T$</td>
<td>Time unit for metric collection and the time window</td>
</tr>
<tr>
<td>$V_{fe}, V_{sc}$</td>
<td>Front-end service and root cause services</td>
</tr>
<tr>
<td>$AD_s, AS$</td>
<td>The clustering result of $RT_i$ and the anomaly score</td>
</tr>
</tbody>
</table>

IV. APPROACH DESIGN

This section introduces the detail of the proposed root cause analysis approach.

A. Overall Framework

To address the above issues, we propose a novel root cause analysis approach named AAMR (short for Automated Anomalous Microservice Ranking). Fig. 1 shows the overall framework of AAMR, which consists of five stages:

$S_1$: Collect system and application-level metrics as the input;

$S_2$: Detect anomalies;

$S_3$: Construct the service dependency graph;

$S_4$: Update the anomaly weights in SDG;

$S_5$: Rank the anomalous nodes and metrics.

$S_1$ and $S_2$ run continuously by default. Once anomalies are detected, the following steps are triggered. We discuss the components of AAMR in detail in the following parts.

B. Data Collection

Root cause analysis is based on performance metrics obtained by monitoring applications. Since a single metric is insufficient to reflect the anomalous degree [7], similar to [4, 5,
we collect metrics at different levels: (i) **System-level Metrics**. These metrics are resource utilization metrics monitored at the physical server or virtual machine layer (e.g., CPU, memory, and network utilization of the host node). (ii) **Application-level Metrics**. Application-level metrics include performance metrics observed at the application layer, such as response time, workload, and network connection.

---

**C. Anomaly Detection**

Anomaly detection is the beginning of root cause analysis. We use the BIRCH [22] clustering algorithm for anomaly detection, which is simple but effective. We continually monitor the response time of each microservice by default. BIRCH takes the RT, collected of each microservice in T as input. As a result, the RT, is divided into n clusters without predefined. It is noticed that the response time of different microservices varies with different business processes. For example, if V_a handles a single business process and V_b handles compound business processes. The response time of V_a is shorter than V_b in most cases. So we cluster RT, for each microservice instead of overall microservices. If the cluster result ADs of a microservice exceeds 1, it indicates this node is anomalous. Instead of simply detecting anomalies [8], we further define the anomaly score (AS) of this node as ADs-1 to represent the basic anomalous degree of each microservice.

**D. Service Dependency Graph Construction**

We construct a service dependency graph based on the network connection between services to represent the anomaly propagation. If service V_a sends a connection request to service V_b, we add a directed edge from V_a to V_b. As for duplicate edges, only one connection is counted to avoid redundancy. By integrating all network connections, we end up with a service dependency graph G(V, E, W). It is a weighted DAG (Directed Acyclic Graph) that describes the dependency between services. Here V, E, W indicate microservice nodes, SDG edges, and the anomaly weights, respectively. Considering that some microservice connections may fail due to anomalies at the current moment, we choose the network connection details from the moment before time window T for the SDG construction.

---

**E. Automated Anomaly Weight Updating**

Once the SDG is constructed, the following processes start to locate the root cause. According to the mRank algorithm [9], if there is an anomalous node in the service network, then the nodes associated with the anomalous node are likely affected. However, it is also possible that other nodes cause the anomalies of these nodes. Therefore, to infer the possibility of a node being abnormal, we need to consider the nodes related to its neighbors. We define AAN(V_i) as the anomalous-adjacent node of node V_i. Further, we define NHAN(V_i) as the next-hop-anomalous nodes of node V_i, that is, the anomalous nodes that directly connect to AAN(V_i). For example, for node A in Fig. 2, AAN(A) consists of B, D, E, and F. And NHAN(A) includes all the anomalous nodes that are connected to B, D, E, and F. Then we define two measurements to quantify the anomaly of a node in the following.

**Definition 4.1 (iScore).** iScore of a microservice V_i in SDG is defined as:

\[
iScore(V_i) = \frac{\sum_{j=1}^{N} AS(V_j)}{\text{Degree}(V_i)}, V_i \in AAN(V_i),
\]

where AS(V_i), Degree(V_i), and N denote the anomaly score of V_i, the degree of V_i, and the number of AAN(V_i), respectively. As for NHAN(V_i) we define:

**Definition 4.2 (xScore).** xScore of a microservice V_i in SDG is defined as:

\[
xScore(V_i) = x(V_i) - \frac{\sum_{j=1}^{N} AS(V_j)}{\sum_{j=1}^{N} \text{Degree}(V_j)}, V_j \in NHAN(V_i),
\]

where x denotes the average anomaly score of NHAN(V_i). Here iScore indicates the anomalous degree of AAN(V_i), and xScore reflects the normality of NHAN(V_i). We count the redundant AS(V_i) and Degree(V_i) only once. For example, in Fig. 2,
iScore(A), x(A), and xScore(A) are 1.5, 1.67, and 1, respectively. Then we define iScore(Vi) as:

\[ iScore(V_i) = iScore(V_i) + xScore(V_i). \] (3)

Clearly, iScore(Vi) is used to combine the multiple pieces of evidence with node Vi itself and its neighbors. If most neighbors of node Vi are anomalous and most neighbors of its AAN(Vi) are normal, node Vi is more likely to be the root cause.

In addition, as presented in [8], the resource utilization of host node hi and the response time of deployed microservices (e.g., Vi) on hi are correlated. For simplicity, we calculate the correlation between the response time metrics of Vi (|Mfe|) and system utilization metrics of hi (|M|) as follows:

\[ \text{Corr}(V_i, h_i) = \frac{\sum_{t=0}^{T} (|M_{fe} - |M_{fe}|)(|M_i| - |M_i|)}{\sqrt{\sum_{t=0}^{T} (|M_{fe} - |M_{fe}|)^2} \sqrt{\sum_{t=0}^{T} (|M_i - |M_i|)^2}}. \] (4)

This correlation function is the Pearson correlation coefficient between the metrics of Vi and hi. The value falls in [0, 1]. In normal cases, the correlation between Vfe and hi is closer to 0. Besides, the system utilization of hi such as CPU, memory, I/O, and network utilization are ranked as the second phase ranking. The max value of Corr(Vi, hi) indicates the key anomalous metric. Finally, the anomaly weight w of Vi can be updated as:

\[ w(V_i) = iScore(V_i) \times \max \text{Corr}(V_{fe}, h_i). \] (5)

Each time an anomaly is detected based on real-time metrics, the anomaly weight for each microservice in the SDG is recalculated for automatically updating. As shown in Fig. 3, the composition of w is the final anomaly weights W in the SDG. Then we normalize W for the random walk algorithm.

With PPR, we get the ranking list of root causes as the first phase ranking. Then we associate the root causes with the anomalous metrics ranking (the second phase) to get a two-phase ranking list, which helps developers mitigate the microservice failures, as shown in Fig. 1(e).

V. EXPERIMENTS

In this section, we conducted experiments to compare our method with several state-of-the-art techniques. The experiments were designed to answer three research questions:

- **RQ1:** Does the proposed method outperform the state-of-the-art approaches in terms of different anomaly cases?
- **RQ2:** Is our approach effective enough to locate the root cause with fast speed?
- **RQ3:** Can our approach adapt to large-scale systems?

A. Setup

1) Experiment Settings. We evaluated the prototype of AAMR on two physical servers. Each physical server has an 8-core 2.4GHz CPU, 16GB of RAM, and Ubuntu 16.04 OS. And we installed Kubernetes 1.13.1, Istio1 1.4.5, Node Exporter2 1.41, and Prometheus3 6.3 on these servers for environment configuration. We used one server to run our system and another server to simulate the workload.

2) Benchmark. The benchmark of experiments is an online shop microservice system named Online-boutique4, which contains 11 microservices. Particularly, since these microservices are mocked and a microservice is used for load generation, effects on these microservices are rather low, and we deployed them on the Kubernetes clusters but excluded them from the evaluation.

<table>
<thead>
<tr>
<th>TABLE II.</th>
<th>WORKLOAD GENERATION DETAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MS</td>
<td>cart</td>
</tr>
<tr>
<td>users</td>
<td>100</td>
</tr>
<tr>
<td>rate(s)</td>
<td>30</td>
</tr>
</tbody>
</table>

3) Data Collection. The workload was generated by Locust5, a distributed load testing tool that simulates concurrent users in an application. Considering real user scenarios, we simulate different request rates for different microservices as shown in Table II. For system-level metrics, we used Node Exporter to collect CPU, memory, I/O, and network utilization metrics. And we used Prometheus, an open-source monitoring tool, to collect response time metrics. These metrics are collected at five-second intervals, and T is set as 150 seconds.

4) Fault Injection. To simulate real-world performance issues, we injected the following three types of failures: (i) Latency Delay. We used the feature of Istio to add a virtual service to instances, which has the effect of increasing the

---

1. https://istio.io
2. https://github.com/prometheus/node_exporter
3. https://prometheus.io
5. https://locust.io
response time of a specified instance to 300ms. (ii) CPU Hog. The performance issue may be caused by the insufficient CPU allocated to the host. We used stress-ng\textsuperscript{\dagger} to stress the system CPU to 99% usage. As for container CPU usage, we limited the utilization of the injected instance by setting Kubernetes configurations. (iii) Container Pause: The “docker pause” command triggers a pause operation on the specified container. The container cannot be shut down directly because of the protection mechanism of Kubernetes.

5) Evaluation Metrics. To quantify the performance of each algorithm, we adopt the same evaluation metrics defined in [6]:

- Accuracy at top $k$ (AC@k) indicates the probability that the top $k$ on the ranking list hits the real root cause for all given anomaly cases. A higher AC@k score represents the algorithm identifying the root cause more accurately. In experiments, we choose $k=1$ and 3. Let $R[i]$ be the rank of each cause and $V_r$ be the set of root causes. AC@k is defined on a set of anomalies $A$ as:

$$AC@k = \frac{1}{A} \sum_{a \in A} \sum_{i \leq k} \left( \frac{R[i] \in V_r}{\min(k, |V_r|)} \right)$$  \hspace{1cm} (6)

- Average accuracy at top $k$ (Avg@k) quantifies the overall performance of an algorithm, where $n$ is the number of microservices. It is defined as:

$$Avg@k = \frac{1}{A} \sum_{a \in A} \sum_{1 \leq i \leq n} AC@k$$  \hspace{1cm} (7)

6) Baseline Methods. To evaluate the performance of AAMR, we compared it to the following baseline methods:

- Random Selection (RS): Random selection randomly selects the possible anomalous microservices among all nodes without any domain knowledge.

- Microscope: Microscope [5] is a graph-based method to locate root causes. For Microscope implementation, we used the 3-sigma principle to detect anomalies and then added these anomalies into a candidate group. We collected the response time for calculating the similarity and ranking the anomalies in the candidate group.

- MicroRCA: MicroRCA [8] extracts an anomalous subgraph based on the SDG. For root cause localization, MicroRCA uses a Personalized PageRank algorithm, which is extended in our approach. To implement MicroRCA, we clustered the RT$_i$ of microservices to extract the subgraph of anomalous nodes.

B. RQ1: Performance Comparison

We tested the performance of AAMR for different fault injection cases. Table III shows the performance in terms of AC@1, AC@3, and Avg@3 for all methods. We can observe that AAMR outperforms the baseline methods in most cases. In 10-round experiments, AAMR achieves an accuracy of 91% for AC@1 and 94% for Avg@3 on average, which outperforms the state-of-the-art methods. The result shows that AAMR gets 3.2% and 9.0% improvement than MicroRCA and Microscope for AC@3, respectively. It is also noticed that the experimental result of the CPU hog case is not as good as other cases because only computation-sensitive microservices are affected in the CPU hog case, e.g., the checkout service and recommendation service in Online-boutique.

<table>
<thead>
<tr>
<th>TABLE III. PERFORMANCE COMPARISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td>Overall</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Latency</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>CPU Hog</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Container</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

\textsuperscript{\dagger}https://kernel/ubuntu.com/cking/stress-ng

In Fig. 4, we compared the performance of each method on different microservices. The result shows that AAMR outperforms other methods in most fault injection cases. MicroRCA performs better in some CPU hog cases because it calculates the correlation between the anomalous node and the host node, which is more accurate but has a higher overhead. However, AAMR performs better on average.
C. RQ2: Localization Time Comparison

Besides accuracy, developers expect to locate anomalies quickly. We set all methods running continuously, and only the top 1 ranking hits the root cause three times consecutively is considered successful. Table IV shows that the execution time of locating the root cause varies from methods, and AAMR takes less time to locate the root cause, i.e., 78% and 72% faster than Microscope and MicroRCA. Here the RS method is excluded in the comparison because of low accuracy.

<table>
<thead>
<tr>
<th>TABLE IV. LOCALIZATION TIME COMPARISON</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
</tr>
<tr>
<td>--------</td>
</tr>
<tr>
<td>MS</td>
</tr>
<tr>
<td>MicroRCA</td>
</tr>
<tr>
<td>AAMR</td>
</tr>
</tbody>
</table>

D. RQ3: Scalability Comparison

Scalability is the main feature of microservice systems. It is noticed that scaling out service replicas will increase the size of the SDG and make it more complicated to locate the root cause. We evaluated the impact of scaling out replicas from 1 to 10 for each microservice in Online-boutique. Fig. 5 shows that AAMR consistently maintains an accuracy of 82-91% for AC@1, which is higher than the state-of-the-art methods.

![Figure 5. Comparison of scalability](image)

VI. CONCLUSION AND FUTURE WORK

In this paper, we design a root cause analysis approach named AAMR. We extend the mRank algorithm to measure the anomaly weight of a node based on its adjacent nodes. After detecting the anomalies by a simple but effective clustering method, we give a two-phase ranking, which helps developers quickly diagnose the system failures. Experiments show that AAMR has an accuracy of 91% and an average accuracy of 94%, which outperforms the state-of-the-art methods.

In the future, we plan to cover more anomaly patterns by adding more metric types. Besides, we will try injecting more faults to test the performance of AAMR in case that multiple anomalies occur at the same time.

VII. ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China (No. 2018YFB1402800) and the National Natural Science Foundation of China (Nos. 62032016 and 61832014).

REFERENCES

Abstract—The anomaly detection based on rich and descriptive system logs is critical to securing information systems. Existing techniques rarely consider semantic information of logs in the detection, resulting in their incapability to handle unseen log events, neither further improve their detection rates. This paper proposes a CNN and LSTM based anomaly detection approach. It utilizes the meaning of log entries — the semantic information of logs in the detection, where the relations among short sequences are automatically learned. The results of comparative experiments demonstrate the effectiveness of the proposed approach on both stable (fixed format) and unstable (unseen, unixed format) logs.

Index Terms—anomaly detection, log analysis, deep learning

I. INTRODUCTION

Large-scale systems, such as cloud-based online service systems, play a core role in a various critical industries. However, as the large-scale systems get increasingly larger and more complex than ever before, they face more security threats. Small system anomalies could lead to huge losses. In order to build a secure system, accurate and timely anomaly detection is needed.

System log is an important resource for troubleshooting and diagnosis of systems. Since system log contains rich and descriptive information, such as timestamp, system states and events, engineers can examine recorded logs for anomaly detection. Traditionally, engineers search keyword in system logs to find indicative anomalies, such as ‘error’, ‘fail’, etc. However, in large-scale system, a great number of logs are generated every day. It is impracticable and time-consuming to diagnose problems through logs manually. Furthermore, both normal and abnormal logs contain words such as ‘fail’ and ‘error’. Therefore, using only keywords search will cause many false positives.

In recent years, many automated log-based anomaly detection approaches have been proposed [1]–[5], some methods [2], [3], [6] treat anomaly detection as a binary classification. In general, these approaches convert logkey (log event indexes) sequences into log count vectors, and then apply data mining methods to detect anomalies. With the prevalence of deep learning, those methods [1], [7] have widely used to detect anomalies. However, the above methods ignore the semantic meaning in the log sequences, and not consider the instability in real-world log. In recent studies, some semantic-based methods have been published [8], [8]–[10]. LogAnomaly [9] proposes the template2vec method, which encodes log template to a semantic vector, and utilizes LSTM to train anomaly detection model. Even though they get good results in certain scenarios, the performance of their methods could be further improved.

The log based anomaly detection faces the following challenges.

1) Semantic meaning of system logs is not utilized well in the detection. Some existing detection techniques do not take semantic information into consideration, while others only calculate semantic information based on all words in the log entry, ignoring that some words have no semantic meanings or even reduce real meanings to some extends. For example, an entry in HDFS logs is “10.250.11.100:50010 Served block blk_3544583377289625738 to /10.250.19.10”. It contains words ‘served block to’, where the word ‘to’ has little meanings in fact.

2) Log is unstable in real-word [10], in other words, many unseen logs are existing in the information systems. In the process of software developers insert or delete certain words to update the logging statements, lots of unseen log events will be generated. Normally the existing techniques only use known logkeys and ignore unseen log events, where unseen log events is only treated as new events. Therefore, they fail to work with unseen log events and cause many false positives.

To address the above challenges, we propose an unsupervised anomaly detection method, which can detect anomalies automatically and achieve high detection accuracy. The proposed method encodes logs into fixed-length semantic vectors. Two logs with similar meanings have similar semantic vectors. Note that a LSTM itself cannot extract enough features to make a good detection, the proposed method employs a CNN and LSTM combined model to detect anomalies. It has the advantages of automatically extracting log short sequence relationship and log quantitative features simultaneously.

We evaluated our proposed method on stable and unstable log dataset. The experimental results show that the proposed approach outperforms the state-of-art techniques with 98% F-measure and only uses 1% normal datasets for training. We also evaluated the proposed method on unstable log dataset. The experimental results demonstrate the effectiveness of the proposed method.
The main contributions of this paper are as follows:

- A novel semantic information embedding technique is proposed to detect anomalies in system logs. The basic idea of extracting semantic information in the detection comes from that system administrators usually use keyword search to find anomalies. Thus, some keywords in the log entries may represent the meanings of the entire log entries.
- A CNN and LSTM combined detection approach is proposed. It has the capabilities of not only learning semantics from system logs, but also learns the quantitative feature from log count vector. It can achieve high detection rates on both stable and unstable logs.

The rest of this paper is organized as follows: Sec. II introduces related work. Then, Sec. III describes the proposed method, and the experiment results are discussed in section IV. Finally, in Sec. VI we conclude our work.

II. RELATED WORK

Logs contain abundant information, such as computer status, making them a valuable resource for anomaly detection. A significant amount of studies have been published in log-based anomaly detection [4], [5]. The current methods are mainly divided into four categories: rule-based methods, data mining-based methods, deep learning-based methods and NLP-based methods.

Rule-based methods use keywords or regular expressions to find anomalies in logs. Marcello [5] proposes a rule-based methods to analyze software failures from logs. Rule-based methods have high accuracy, but they require domain expertise and are very time-consuming in production environment. In the past few years, data mining and deep learning methods have been proposed.

There are many methods based on data mining [2], [3], [6]. These methods generally first parse log messages into logkey. Then they convert the logkey sequences into a log count vectors, and finally apply data mining methods to detect anomalies. Data mining methods can be divided into unsupervised methods and supervised methods. Compared with unsupervised methods, supervised methods have better results. However, supervised method requires a large quantity of labeled data, which need massive manual effort in large system. Furthermore, in real environments, abnormal data is rare compared with normal datasets, which make it impractical for supervised method training. Therefore, our method only uses normal logs for training.

In recent years, deep learning-based methods have been widely studied [1], [7]. Deeplog [1] uses normal datasets to train an anomaly detection model. It uses LSTM to predict the next logkey and compares it with the actual logkey to detect anomalies. These methods do not consider the semantic information in log entries, and they have poor performance on the unseen log data. In real-word log data, logs are unstable because the evolution of logging statements and noise generated during log data pre-processing [10]. For example, developers often add or delete certain words when they update a logging statement. Many unseen log events will be produced. Existing approaches have poor performance on this issue due to the wrong classification. DeepLog uses a user feedback mechanism to update anomaly detection models. However, it needs lots of manual feedback which is infeasible in real-time systems. We propose a method to handle with unseen log events without the need of manual feedback.

The latest studies [8]–[10] use NLP techniques to analyze log-based anomaly detection. LogAnomaly [9] extracts the semantic information form log, and then use LSTM to train the anomaly detection model. LogRobust [10] can also represent log message as semantic vectors, and utilizes the attention-based Bi-LSTM model to detect anomalies. However, these methods only use LSTM to train anomaly detection models. Our method combines CNN and LSTM, which can extract the richer features.

III. METHOD

A. Overview

The overview of the proposed method is shown in Fig. 1, which mainly includes two parts, log vectorization and anomaly detection model. First, we leverage Drain [11] to parse the log into logkey and group logkey by identifiers(such as block_id in HDFS dataset). After that, the proposed method does not rely on logkey sequences($L_{sequences}$) for anomaly detection like exiting methods. Instead, it encodes logkey sequences into semantic vector sequences($S_{semantic}$) and log count vector($V_{count}$). Then, we propose an anomaly detection model that can detect semantic vector sequences and log key count vectors simultaneously. The model has the ability to extract short sequence relationship among log entries, which is suitable for analyzing log sequences data. In the end, an
alarm message could be sent to system administrator if an anomaly is detected.

B. Semantic Embedding

The proposed method can extract semantic information from log entries and encode each logkey into a fixed-dimension vector. The design of converting logkey into a semantic vector is based on the assumption that some keywords in a log entry can represent the meaning of whole log entry. For example, a log entry “081109 203521 145 INFO dfs.DataNode$DataXceiver: Receiving block blk_...” from HDFS dataset, which contains rich information, such as timestamp, pid, IP address, block_id etc. The meaning of this log entry can be represented as two keywords “receiving block”. The manual effort to find keywords is impractical in large system, so we need an automatic method to determine which words in a log entry are keywords and how to convert the keywords to a semantic vector. As shown in Fig. 2, the workflow of semantic embedding consists of three phases, keyword search, word vectorization and concatenation.

1) Keywords Search: Get keywords from the log template. First we need pre-processing. In this step we filter non-words(such as ‘*’, ‘:’) in the template entry, and split the template into individual word. Then, we use TF-IDF [12] to calculate the importance of each word in the template entry, which can effectively measure importance of words in a document. For each word in template entry, its TF-IDF weight is calculated by TF*IDF. The term frequency(TF) represents the frequency of words in the template entry. For example, the word ‘block’ appears multiple times in the template, which means that word ‘block’ has a high TF weight. However, if the word ‘block’ appears in all template entries, it means that it can not distinguish between those template entries, so its weight should be reduced. Therefore, we also calculate the inverse document frequency(IDF). If the word appears multiple times in the template entry, it has low IDF weight. We calculate words importance by the following formula.

\[ TF = \frac{N_{word}}{N_{word\_total}} \]  
\[ IDF = \log \left( \frac{N_{template}}{N_{template\_total}} \right) \]  

where \( N_{word} \) represents the number of target word in the log template. \( N_{word\_total} \) is the total number of word in a log template. \( N_{template} \) is total the number of log template containing target words. \( N_{template\_total} \) is the total number of log template. After getting the TF-IDF weight of each word, we sort the word importance according to their TF-IDF weight. And we set the parameter \( g \), which represents the number of keywords used. As the Fig. 2 shows, \( g=3 \) and we can get top 3 keywords in the template.

2) Word Vectorization: In this phase, we convert each word in log into a semantic vector. We use Glove [13] algorithm encodes each word to a word semantic vectorwhich map each word to a fixed dimension vector. Two words that are semantically close have similar word semantic vector.

3) Concatenation: As we get keywords and word semantic vector, we can convert log into a semantic vector. As the Fig. 2 shows, we convert the keywords into word vectors according to the Glove model, and then we concatenate the word vectors to get semantic vector. The dimension of semantic vector is \( w*g \), in which \( w \) is the word semantic dimension and \( g \) is the number of keywords.

C. Model

Our anomaly detection model combines CNN and LSTM. CNN is used for capture patterns from semantic vector sequence(a list of semantic vector), because our CNN model has strong ability to extract the short sequence relationships. As for LSTM [14], it can learn the quantitative patterns from log count vector. The combination of CNN and LSTM model can improve the accuracy, which is demonstrated in Sec. VI.

Due to the fact log is a kind of text, it can take a benefit of natural language processing (NLP). Our CNN model refers to the sentence classification model in the NLP field [15]. The CNN neural network contains several layers. The input layer we use semantic vector sequence as input, which is a \( m*n \) matrix, where \( m \) represents the size of the sliding window, and \( n \) represents the dimension of the semantic vector. The next layer is CNN convolutional layers, which is the core layer of CNN. It uses three different one-layer filters to convolute over the input layer. These filters have same width but different heights. The filters width is the same as the dimension of semantic vector, so these filters can only move in the height direction. And then we use 1-max-pooling layers to obtain the maximum feature from feature vector. In this way, we get the semantic vector sequence feature.

LSTM is a kind of recurrent neural networks that designed for sequential data. Thus, we use LSTM neural network to capture quantitative patterns from log count vector. The LSTM neural network consists of an input layer, a hidden layer and an output layer. The LSTM unit to calculates the new state and output uses the input data and the previous unit state. A series of LSTM unit form an LSTM neural network.

Finally we concatenate the CNN and LSTM feature map, and add a softmax function in the output layer. The softmax function outputs the probability of the next logkey.
D. Detection

The proposed method uses the normal execution path to train the anomaly detection model. If the log execution path deviates from the model prediction, we can send an alarm message. The Logkey sequence represents the execution path of log. Let \( k=(k_1, k_2, k_3 \ldots k_n) \) as the whole set of distinct logkey and there are \( n \) different logkeys. The main idea is using the most recent \( m \) logkey to predict the next \( m+1 \) logkey, so we treat anomaly detection as a multi-classification problem, where each logkey is a class. We use a sliding window of \( m \) to split the logkey sequence. A subsequence is obtained as \((k_j, k_{j+1} \ldots k_{j+m−1})\). The next logkey is \( k_{j+m} \). For example, there has a logkey sequence \((1,2,3,4,5)\). First we set a sliding window size as \( 3 \). Dividing the log sequence according to the sliding window, we can get logkey subsequences as \((1,2,3),(2,3,4)\) and their next logkey is \( 4,5 \) respectively.

The proposed method converts logkey sequence to semantic vector sequence and log count vector. The log count vector is wildly used in anomaly detection. It represents the number of occurrences of each logkey in a sliding window. The entire logkey can be expressed as \( k=(k_1, k_2, k_3 \ldots k_n) \) and there have \( n \) distinct logkey. Thus, the log count vector is \( n \) dimension. Log count vector is denoted as \((c_1, c_2 \ldots c_j \ldots c_n)\), where \( c_j \) is the number of \( j \)-th logkey in the sliding window. Finally, the proposed method inputs the semantic vector sequence and log count vector into the anomaly detection model.

In anomaly detection, taking the most recent logkey as input value, the anomaly detection model returns a prediction result, that is, the probability of next logkey. The proposed method select top \( g \) probabilities as candidates. If the next logkey is not in the top \( g \) candidates, it can be considered abnormal.

IV. EXPERIMENT

A. Dataset

1) Stable Dataset: We conduct our experiment on stable log dataset HDFS [2] and BGL [16]. Tab. 1 shows summary of stable log data set.

   The HDFS dataset is a benchmark dataset for log anomaly detection. It is generated by Hadoop-based map-reduce jobs on Amazonas EC2 with more than 200 nodes. In total, 11,197,954 log messages are collected. Since HDFS data set is labeled by block_id, we use block_id as identifier to group log entries and get 558,221 normal sessions and 16,838 abnormal sessions. The data set is unbalanced, only 2.9% of the datasets are abnormal sessions. The proposed method only uses less than 1% normal sessions for training, which is a grouping of the first 100,000 log entries of original dataset.

   BGL is a open source dataset used in Blue Gene/L supercomputer system at Lawrence Livermore National Labs. The BGL dataset contains 4,747,963 logs, in which each log is labeled as abnormal or normal, and 348,460 logs are labeled as abnormal. We use a window size of 10 to slice logs into log sequences, and randomly take 20% for training the others for testing.

2) Unstable Dataset: In order to evaluate the robustness of the proposed method. We create unstable log event datasets based on the BGL dataset, with randomly adding or deleting words in the BGL dataset as shown in Fig. 3.

B. Parameter Setting

Our proposed method uses the following parameters: \( w=10 \), \( g=8 \), \( k=(2,3,4) \), \( t=4 \), \( l=2 \), \( m=64 \), \( n=150 \). \( w \) is the window size, and \( g \) is the number of candidates. If the next logkey is in the top \( g \) of the prediction candidates, it is considered normal. \( k \) is the convolution kernel of CNN. For example, \( k=(2,3,4) \) means that there have three different convolution kernels and their heights are 2, 3, 4 respectively. \( n \) and \( t \) denote the number of feature maps of CNN and the number of keywords to represent a log entry. \( l \) and \( m \) is the number of layers and number of neurons in LSTM. For other methods, we use the parameters with their best results.

C. Evaluation Metrics

In order to measure the prediction accuracy and recall rate, we introduce four indicators: true positive (TP) is anomalous block predicted to be anomalous, false positive (FP) is normal block predicted to be abnormal, true negative(TN) is normal block predicted as normal, and false negative (FN) means an abnormal block predicted to be normal.

The calculation formulas of precision, recall and F-measure are as follow.

\[
\text{Precision} = \frac{TP}{TP + FP} \tag{4}
\]

\[
\text{Recall} = \frac{TP}{TP + FN} \tag{5}
\]

\[
F – \text{measure} = \frac{2 \times \text{Precision}}{\text{Precision} + \text{Recall}} \tag{6}
\]

D. Comparison

We compare the proposed method with four unsupervised baseline methods PCA [2], LogCluster [6], DeepLog [1] and LogAnomaly [9].

1) Compared The Proposed Method with Exiting Methods on stable dataset: Fig. 4 shows the comparison of results
of the proposed method and the other four baseline methods on HDFS dataset. Apparently, the proposed method achieves best result, with F-measure 0.98. Both PCA and LogCluster methods achieves good results on precision, with the price of low recall. The state-of-art method LogAnomaly encodes log template to semantic vector for anomaly detection, and has better performance compared with DeepLog. However, the performance of LogAnomaly could be further improved. The proposed method gets F-measure 0.98, which higher than LogAnomaly with F-measure 0.97.

Fig. 5 shows the performance of our method and other four baseline method on BGL dataset. Among those methods, our method achieves the best F-measure 0.954, and LogAnomaly has an F-measure 0.945 as second. The reason why our method has better performance than LogAnomaly, is as follow. First, LogAnomaly extracts the semantic vector based on all words in the log entry, but our method only considers the keywords in log entry to calculate semantic vector. In addition, LogAnomaly uses LSTM model to extract patterns from logkey sequence, which can not extract log short sequence relationship well. Our proposed method uses a LSTM and CNN combined model that has ability to extract log short sequence relationship.

2) Compared The Proposed Method with Exiting Methods on Unstable Log Events: As the log system evolve and log parsing error, many new log events will be produced. These methods such as DeepLog and LogCluster cannot effectively handle these unseen log events.

Our method is based on the assumption that most of the new logs are variants of the original logs and will not change the meaning of the original logs, so the proposed method can match the new logs with the original logs significantly. First the proposed method use Drain extract the new log to the template, then search for keywords in the log based on TF-IDF, and finally calculate its similarity by the following formula:

$$similarity = \frac{2 \times N_{same}}{N_{new} + N_{exist}}$$  

$$N_{same}$$ means the number of keyword both in new template and exist template. $$N_{new}$$ means the number of word in new template and $$N_{exist}$$ means the number of word in exist template. After obtaining the similarity between new template and exist template, the proposed method can match new template to an existing one.

We evaluated the proposed method on unstable BGL log dataset. The experimental results on the unstable log event datasets are shown in Tab. II. The injection ratio represents the ratio of the number of randomly adding and deleting logs to the total number of logs. Note that PCA is omitted from this table because of its very poor performance. Clearly, our method has achieved the best performance. As the injection ratio increases, the performance of our method decreases slowly (F-measure from 0.95 to 0.94). And the performance of the DeepLog method has declined a lot (F-measure from 0.93 to 0.42). LogCluster achieves better F-measure than DeepLog with F-measure declined from 0.86 to 0.76 as the injection ratio increases. The reason is that LogCluster and DeepLog treat unseen log events as a new log events, which may cause false alarms.

E. Discussion

1) Impact of CNN and LSTM in Proposed Method: Our CNN and LSTM combined model can extract patterns from semantic vector sequences and log count vectors simultaneously. Tab. III, demonstrates the impact of CNN and LSTM
in proposed method. We calculate the precision, recall and F-measure of the proposed method without (w/o) LSTM, and the proposed method without (w/o) CNN. The proposed method without CNN has a much low F-measure, which demonstrates semantic vector sequences are important for the anomaly detection model. The proposed method without LSTM has a lower F-measure. By combining CNN and LSTM model, the proposed method obtains best results.

2): Impact of Keyword Numbers in Proposed Method: In Sec. IV, we describe the semantic vectorization, which can capture the semantic information from log. We want to verify the impact of keyword numbers on the accuracy of the proposed method. We consider 6 distinct numbers from 1 to 6. We follow the same parameters and utilize the same training and testing dataset.

Intuitively, we think that the more keywords are used, the more semantic information will be captured from the log. However, Tab. III shows as the number of keywords increases. The value of F-measure first increases and then decreases, the value of F-measure is the highest when the number of keywords is 4. This is mainly because as the number of keywords increases, some unimportant words(such as ‘of’, ‘the’) will also be converted into semantic vectors, which may add noise to semantic vector and reduce the F-measure.

V. CONCLUSION

To address the challenges caused by unstable system logs, this paper proposes a CNN and LSTM based anomaly detection approach. The proposed approach can automatically learn the semantic information among system log sequences and embed the semantic information in the detection. As a result, the proposed approach obtains high detection rates on both stable and unstable logs, in comparison to the existing methods in the experiments.

However, our present method has a major limitations. The proposed approach gets better results than exiting methods, with the price of it takes more time for anomaly detection. One of the future directions of our work is to reduce the time for model anomaly detection.

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>THE IMPACT OF CNN AND LSTM IN PROPOSED METHOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td>Precision</td>
</tr>
<tr>
<td>without (w/o) CNN</td>
<td>0.91</td>
</tr>
<tr>
<td>without (w/o) LSTM</td>
<td>0.96</td>
</tr>
<tr>
<td>CNN and LSTM</td>
<td>0.97</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE IV</th>
<th>THE IMPACT OF KEYWORD NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of keywords</td>
<td>Precision</td>
</tr>
<tr>
<td>1</td>
<td>0.963</td>
</tr>
<tr>
<td>2</td>
<td>0.965</td>
</tr>
<tr>
<td>3</td>
<td>0.963</td>
</tr>
<tr>
<td>4</td>
<td>0.966</td>
</tr>
<tr>
<td>5</td>
<td>0.959</td>
</tr>
<tr>
<td>6</td>
<td>0.960</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENTS

This work is supported by the Key Research and Development Program of Guangdong Province (Grant No. 2019B010136001) and the National Natural Science Foundation of China (Grant No. 61672494). The corresponding author is Shuyuan Jin.

REFERENCES

Abstract—A secure design pattern is a well-proven reusable solution to a recurring security problem that arise in specific contexts. Using secure design patterns properly can help tackle software vulnerabilities during software development. However, the lack of selection guidance of secure patterns makes it more difficult for developers to use secure design patterns than conventional design patterns. To address this issue, this paper presents a methodology of selecting the appropriate secure design patterns for software vulnerabilities formalized in anti-patterns. This methodology bridges the gap between the vulnerabilities and secure design patterns to produce a useful tool for secure software development.

Index Terms—software vulnerabilities, anti-patterns, secure design patterns

I. INTRODUCTION

Software vulnerabilities are weaknesses in a system’s architecture, design, or code that can cause violations of the system’s security policy. Its existence is the primary cause of attacks on software systems [1]. Even though security is a critical quality attribute, security is often seen as an afterthought. Many research studies on software vulnerabilities focus on testing methodologies, such as using machine learning [2] to identify the common vulnerabilities.

Although software testing is absolutely mandatory and necessary in locating the vulnerabilities, security needs should be emphasized throughout the entire software development process and the vulnerabilities should be identified as early as possible. After software has been deployed, it becomes more expensive to remove vulnerabilities by patching. Thus, software developers should elicit security expectations during the requirements analysis stage and consider them during the stages that follow. Citing research by Software Engineering Institute (SEI), the U.S. Department of Homeland Security (DHS) states in its Software Assurance information sheet that “90% of reported security incidents result from exploits against defects in the design or code of software” [4]. Most of the vulnerabilities reported in software are the results of bad decisions made during the implementation stage, however, many of them actually originate in the previous software development stage, the design stage. “Secure by design” is not a new concept and has been explored by many researchers.

Deogun et al. [5] introduced the good practices for implementing essential software features using design as the primary driver for security. Several researchers have published various secure design patterns to address security vulnerabilities at the design level [8], [9]. However, very few research studies have explored the methodology of selecting appropriate secure design patterns to mitigate security vulnerabilities.

This research proposes a methodology for selecting the appropriate secure design patterns for addressing software security vulnerabilities. The rest of the paper is organized as follows. Section 2 briefly discusses the security vulnerabilities, secure design patterns, and anti-patterns. Section 3 presents the methodology of selecting appropriate secure design patterns for addressing the security vulnerabilities. Section 4 discusses the related works, and section 5 concludes the work.

II. BACKGROUND

A. Software Vulnerabilities

A software vulnerability is an exploitable flaw in any part of a system’s artifact or component. Cross-site scripting (XSS) is an example of a commonly recurring security vulnerability in web applications. With the advent of scripting languages such as JavaScript, it has become feasible for attackers to inject malicious scripts into a victim’s browser. By injecting the script, attackers can read private data, hijack the user session, or delete essential data. Fig.1, 2 & 3 show an example of a DOM-based XSS attack, which is found in OWASP’s Juice Shop application [13]. The application’s search product function is intentionally developed in such a way that user input is not validated or sanitized before execution. An attacker can enter malicious script (Fig.1) as user input in the search field. The malicious script entered into the search field (Fig. 2) is then submitted to the server and gets executed on the client side. In this example, code from a different domain (soundcloud.com), in the form of a soundtrack, is inserted instead of legitimate search results (Fig. 3).

B. Anti-Patterns

Anti-patterns are commonly occurring solutions to a problem that generates negative consequences [3]. Software development is a complicated process and many decisions may cause a project to fail. Formal documentation of the decisions
Fig. 1. XSS malicious script

Fig. 2. The malicious script is entered into the search field

Fig. 3. The malicious script injects code from another domain when submitted

or processes that lead to failure can guide future software engineers on what to avoid. Anti-patterns can help developers capture the causes of security vulnerabilities in architecture, design, or source code. Dougherty [8] discussed the need for anti-patterns to describe bad decisions, causing software security failures in a formal way. Nafees et al. [12] proposed the format of documenting vulnerability into anti-patterns.

C. Secure Design Patterns

Secure design patterns help to prevent the occurrence of vulnerabilities [8]. Secure design patterns address security issues in the architectural design and implementation phases of the development life cycle [8]. Many secure patterns have been proposed and a good number of significant work have been published [8], [9].

III. METHODOLOGY

The approach we designed for selecting secure design patterns to address security vulnerabilities include 3 steps: 1) formalizing a software vulnerability using anti-pattern description model; 2) selecting the secure design pattern candidates that can address the vulnerability’s anti-pattern; and 3) testing the selected secure design pattern(s) for the anti-pattern problem.

A. Formalizing Vulnerability to Anti-Pattern Description

The first step is to identify and formalize the vulnerability using an anti-pattern description. We adopt the anti-pattern description model [12] with the following modifications:

- Our model follows conventional anti-pattern description model to include elements Context, Problem, Solution, and Consequences. These elements are important to help the audience understand what the actual vulnerability is.
- Our model introduces the element Root Causes, which is an essential factor in identifying appropriate solutions.

The elements Problem, Context, Root causes, and Solution are the major factors to address in mapping the anti-pattern to appropriate secure design patterns. Our vulnerability anti-pattern description model has ten elements. The following is an example of using the anti-pattern description model to specify Cross-Site Scripting (XSS) anti-pattern.

1) **Anti-pattern Name:** Cross-Site Scripting Anti-pattern
2) **Also Known as:** Improper Neutralization of Input During Web Page Generation, XSS
3) **Context:** Any web application that uses JavaScript, VBScript, ActiveX, CSS, or any other scripting language
4) **Anti-pattern problem:** Cross-site scripting (XSS) is an injection attack in which attackers can execute malicious scripts on the victim’s browser.
5) **Root causes:**
   - Lack of input validation: Vulnerable applications trust the data from input without validating it. Without validation, malicious content can be executed in the victim’s browser.
   - Lack of data sanitization: The victim’s browser misinterprets external malicious data as a part of the script and executes it without sanitization.
6) **Example:** A DOM-based XSS attack example is found in OWASP’s Juice Shop application [13], as shown in Section II-A.
7) **Consequences:** Private data breach; User session hijacking; Identity theft; Phishing Attacks; Web site defacement; Port scan; Keylogging; Trojan Attack
8) **Solution:**
   - Data sanitization: Sanitize the external inputs via encoding or escaping. The encoding must be applied to all potential vectors.
   - Input validation: Validate the inputs using blacklist or whitelist validation (preferable).
   - Miscellaneous solutions: Using process level, technology-specific or configurational solutions, such as implementing Content Security Policy, using HTTPOnly cookie flag, SameSite cookie parameter, etc.
9) **Attack Types:** Reflected/non-persistent XSS attack, Stored/persistent XSS attack DOM-based XSS attack, Self-XSS, Mutated XSS, and Universal XSS attack
10) **Common Weakness Enumeration:** CWE-79 [18]

B. Selecting secure patterns for addressing the anti-pattern

The second step of the approach is to go through the pool of secure design patterns and select the best-fit candidates that can address a vulnerability anti-pattern. Two phases, Collection Phase and Analysis Phase, are involved in this step.

1) **Collection Phase:** Secure patterns are often published in conferences, academic literature, books, repositories, and the internet [8], [9]. The collection phase begins with searching
for published secure design patterns. The most crucial problem in the pattern community is, there is no single comprehensive secure pattern repository that exists today. This step can be skipped after a secure pattern repository is developed.

Below are the samples from our collected secure patterns for addressing the XSS anti-pattern. They are categorized into architecture, design and implementation levels.

- Architectural level: Application Firewall, Broker, Roles [15]
- Implementation level: Account lockout, Client Input Filters, Input Validation [8]

2) Analysis Phase: The analysis phase is to analyze the collected patterns from the previous phase. The approach is to find potential solutions by mapping the elements Root causes, Problem, Context, Solution of a vulnerability anti-pattern description to the essential elements of secure design patterns. These elements include Intent, Problem, Context, Forces, Motivation, Applicability, Solution, Structure/Participants, and Collaborations. The following questions should be asked and answered during the mapping process:

1) Question1: Does the Intent element is present or sufficient in capturing the pattern’s purpose?
2) Question2: Does the Intent of the secure pattern help address the vulnerability anti-pattern’s root cause(s)?
3) Question3: In case the Intent element is not present or insufficient in capturing the purpose of the pattern, Do Problem/Context/Forces/Motivation/Applicability are relevant to the vulnerability anti-pattern’s root cause(s)?
4) Question4: Does Solution/Structure/Participants and Collaborations of the secure pattern help deliver the Solution of the anti-pattern?

The workflow of the mapping process is illustrated in Fig. 4. The analysis begins with scanning of the pattern description model’s Intent element to understand the purpose of the pattern and check if it can map to the root cause(s) of the targeted vulnerability anti-pattern. If the Intent element is absent or insufficient, explore other elements (such as forces) of the secure pattern description model. If the secure pattern’s Intent or solution does not address the root cause, the pattern is ruled out; otherwise, add the pattern in the selected pattern list.

<table>
<thead>
<tr>
<th>Secure Patterns</th>
<th>Intent</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Fire-wall [A]</td>
<td>To filter calls and responses to/from enterprise applications, based on an institution access control policy</td>
</tr>
<tr>
<td>Broker pattern [A]</td>
<td>Coordinates communication between client and server via requests and responses</td>
</tr>
<tr>
<td>Roles [A]</td>
<td>Organizing the users with similar security privileges</td>
</tr>
<tr>
<td>Controlled Object Monitor[D]</td>
<td>To control access to objects by processes</td>
</tr>
<tr>
<td>Intercepting Filter [D]</td>
<td>To provide mechanism for centrally intercepting the requests and pass them through series of filters before forwarding them to intended destination</td>
</tr>
<tr>
<td>Secure Chain of Responsibility[D]</td>
<td>To preprocess or postprocess requests/responses using series of handlers</td>
</tr>
<tr>
<td>Secure Logger [D]</td>
<td>To facilitate in centralized logging mechanisms</td>
</tr>
<tr>
<td>Secure Strategy Factory [D]</td>
<td>To facilitate easy creation of security objects and use of interchangeable security strategies</td>
</tr>
<tr>
<td>Account Lockout [I]</td>
<td>Lock the user’s account after limited number of incorrect password attempts</td>
</tr>
<tr>
<td>Client Input Filters [I]</td>
<td>All incoming requests from the client should be filtered at the server.</td>
</tr>
<tr>
<td>Input Validation [I]</td>
<td>To validate all external inputs from untrusted data sources</td>
</tr>
</tbody>
</table>

The mapping process of secure patterns’ Intent to XSS anti-pattern’s root causes are illustrated in Table 1. It lists the Intent of each of the collected secure design patterns from the Collection phase. The intent of each secure pattern is then compared with the root cause of the XSS anti-pattern to find a mapping. Table II gives the reasoning of whether a secure pattern is considered fitting to solve the anti-pattern’s root causes. Several secure patterns are ruled out at this point.

Same strategy is used in the analysis of other collected secure patterns. Due to the page limit, the complete analysis process is not presented. Table III presents the selected secure patterns to address the XSS anti-pattern without considering additional requirements on the solution space, such as performance and flexibility. However, additional requirements or attributes required in the design are not avoidable. They should be carefully analyzed in this phase.

C. Testing the suggested secure design patterns

The final step is to apply the selected secure design pattern on the already vulnerable application to test whether the secure design patterns can tackle the anti-pattern problem. Any false positives from the final list are eliminated at this stage. We redesigned the OWASP Juice Shop [13] with Secure Strategy Factory pattern, Input Validation pattern and Intercepting Filter pattern at the design and implementation level [17]. Manual security testing was performed on [13]. The malicious script shown in Fig. 1 was not executed by browser after the redesign. Results show that the selected secure design patterns can be adopted to address XSS anti-pattern problem.
Addressing root causes of XSS Anti-pattern?

**TABLE II**

<table>
<thead>
<tr>
<th>Secure Pattern</th>
<th>Addressing root causes of XSS Anti-pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Firewall[A]</td>
<td>The pattern supports the usage of firewalls to detect possible attacks by scanning for malicious signatures using Input Validation. It is most relevant to our anti-pattern problem.</td>
</tr>
<tr>
<td>Broker pattern[A]</td>
<td>This pattern may help if a separate component is placed between client and server to validate or sanitize the external data.</td>
</tr>
<tr>
<td>Roles [A]</td>
<td><strong>Ruled out:</strong> The pattern is related to organizing the users with same role, which is irrelevant to the anti-pattern problem.</td>
</tr>
<tr>
<td>Controlled Object Monitor[D]</td>
<td><strong>Ruled out:</strong> This pattern is for controlling objects, which is irrelevant to our anti-pattern problem.</td>
</tr>
<tr>
<td>Interception Filter[D]</td>
<td>The pattern may be used to intercept all web requests and pass them through filters to eliminate malicious requests using Input Validation.</td>
</tr>
<tr>
<td>Secure Chain of Responsibility[D]</td>
<td>May be used for preprocessing the web requests using Input Validation and/or Sanitization.</td>
</tr>
<tr>
<td>Secure Logger[D]</td>
<td><strong>Ruled out:</strong> Logging can not help in addressing the anti-pattern problem.</td>
</tr>
<tr>
<td>Secure Strategy Factory[D]</td>
<td>The pattern allows interchangeable strategies and separate them from client who uses it. May help in designing Input Validation/Sanitization strategies.</td>
</tr>
<tr>
<td>Account Lock-out[I]</td>
<td><strong>Ruled out:</strong> The pattern is related to authentication, which does not help in Input Validation/Sanitization.</td>
</tr>
<tr>
<td>Client Filter Inputs[I]</td>
<td>This pattern may help if the good inputs are filtered from bad inputs using Input Validation.</td>
</tr>
<tr>
<td>Input Validation[I]</td>
<td>This pattern is most relevant to our anti-pattern problem.</td>
</tr>
</tbody>
</table>

**TABLE III**

<table>
<thead>
<tr>
<th>Vulnerability Anti-Pattern</th>
<th>Secure Design Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>XSS Anti-pattern</td>
<td>Broker; Application Firewall</td>
</tr>
<tr>
<td></td>
<td>Interception Filter; Secure Strategy Factory; Secure Chain Of Responsibility; Input Validation; Client Input Filters</td>
</tr>
</tbody>
</table>

IV. DISCUSSION

The research on security pattern is an active and growing field across the globe [14]. Much research has been done on the secure design pattern classification, but little research has considered selecting appropriate secure design patterns for a given problem. The pattern selection approach proposed in [15] was generalized and not intended for security goals. Alvi et al. [1] proposed a security pattern selection technique based on security objectives and security flaws. However, security flaws do not formally capture what bad decisions can cause the vulnerability in the applications. Different from Alvi’s [1] approach, our methodology is based on the anti-pattern model, which presents not only the result but the causes of a vulnerability. Our research does not aim for suggesting the best pattern for a given context like the other approaches [1], [16], but on selecting potential pattern(s) for an anti-pattern problem. In addition to the secure patterns discussed in this paper, other patterns, such as the process patterns and technology patterns [3], can also be used to address vulnerability anti-pattern problems. Our methodology can be easily extended to involve such patterns in the solution.

V. CONCLUSION

Majority of the security vulnerabilities are in software and many security weaknesses in software originate in the design stage during the software development process. It is critical to tackle the vulnerabilities in the software design. This research demonstrates a novel approach of selecting appropriate secure design patterns based on the vulnerability anti-pattern model to mitigate common software vulnerabilities in the design. The future work will be focused on two directions. One direction is to track published up-to-date secure patterns and develop a web-based repository of these patterns for the researchers and developers. Another direction is to develop a recommendation tool that applies this approach to identify the anti-patterns of the top eight most common web application vulnerabilities [11] and recommend appropriate secure design patterns.

REFERENCES

Formal Modeling and Verification of ICN-IoT Middleware Architecture

Hongqin Zhang, Jiaqi Yin, Huibiao Zhu*, Ningning Chen
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China

Abstract—As a key technology of the Internet of Things (IoT), middleware plays an important role in managing virtualized resources and services. However, traditional Internet architectures cannot ensure adequate data security and efficient data delivery for IoT middlewares. Therefore, Information-Centric Networking (ICN), a paradigm of the future network, is introduced into IoT middlewares. Since ICN-IoT middleware is attracting more and more attentions, its security is worth discussing.

In this paper, we adopt Communicating Sequential Processes (CSP) to model the ICN-IoT middleware architecture. Five properties (deadlock freedom, data availability, action keys leakage, device faking and user faking) of the model are verified by utilizing the model checker Process Analysis Toolkit (PAT). According to the verification results, the model cannot guarantee the security of data. To solve the problems, we encrypt messages with the receiver’s public key, and improve the model by introducing a method similar to the digital signature. The new verification results demonstrate that our study can assure the security of the ICN-IoT middleware architecture.

Index Terms—ICN, IoT Middleware, CSP, PAT, Modeling, Verification

I. INTRODUCTION

The Internet of Things (IoT) [1] is an emerging paradigm, which connects heterogeneous devices with the Internet. As a crucial technology of IoT, IoT middleware [2] manages the integration of devices and provide interested users IoT services. The effectiveness of information retrieval and security of the transmitted information are two key challenges of IoT middlewares. To cope with the challenges, several IoT middleware solutions have been proposed [3]–[5]. Park et al. put forward a cloud-based middleware for self-adaptive IoT collaboration services, which improved the feasibility and performance of IoT systems [3]. Sicari et al. proposed a quality-aware IoT architecture aiming to deal with the data security and quality [4]. Shi et al. came up with an SDN-like publish/subscribe middleware architecture [5]. It used a machine learning method based on the eXtreme Gradient Boosting (XGBoost) model to improve the efficiency of IoT systems [5]. However, the above solutions cannot support the effectiveness and security of IoT systems at the same time. Hence, a solution called ICN-IoT middleware architecture [6] was proposed by introducing Information-Centric Networking (ICN) [7] into IoT. ICN identifies a network object by the name instead of the IP address, which supports content-oriented security and effectiveness [8]. Whereas, there are few works on the verification of the ICN-IoT middleware architecture.

*Corresponding author: hzh@sei.ecnu.edu.cn (H. Zhu).

In this paper, the ICN-IoT middleware architecture is formally modeled using the process algebra CSP [9]. Model checking tool PAT [10] is adopted to verify its functional and security properties. The verification results demonstrate that the architecture may cause action keys leakage and device faking. Therefore, we improve the original architecture by encrypting the messages with the receiver’s public key to protect action keys, and introduce a method similar to digital signature to avoid device faking. Then we verify the improved architecture using PAT. The new verification results show that our work can enhance the security of the architecture.

The rest of this paper is organized as follows. Section II briefly introduces the ICN-IoT middleware architecture and CSP. Section III is devoted to the modeling of the ICN-IoT middleware architecture. In Section IV, we analyse the verification results and give the improvement that can address the vulnerabilities of the architecture. Finally, conclusions and future work are given in Section V.

II. BACKGROUND

In this section, we give a brief description of the ICN-IoT middleware architecture. After that, we introduce the syntax of the process algebra CSP.

A. ICN-IoT Middleware Architecture

ICN-IoT middleware architecture is designed to build a unified IoT platform using ICN. The schema of the architecture is illustrated in Fig. 1.

![Fig. 1: ICN-IoT middleware architecture (simplified from [7])](image)

The architecture involves five entities:

- **Device**: It collects data from the environment and publishes them to the aggregator.
- **Aggregator**: It deals with the data received from devices. For simplicity, we use $Agg$ to represent the aggregator.
• **Local Service Gateway (LSG):** It serves to connect the local IoT system to the global one and handle the local name assignment.

• **ICN-IoT Server:** It manages the subscriptions within the IoT system, provides subscribers services and enforces data access policies.

• **User:** The user interacts with the ICN-IoT server to get the data for subscribed services.

The core functions supported by the architecture are: (i) device discovery aiming to connect a new device with the system and establish relationships between nodes; (ii) service discovery meaning to subscribe to IoT services; (iii) naming service denoting assigning persistent names to devices; (iv) content delivery representing forwarding data to subscribers.

The related notations and descriptions are listed in TABLE 1.

### TABLE 1: Notations and descriptions

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$pu_k$, $pr_k$</td>
<td>Public/Private key of the device/user/intruder, $x \in {d, u, i}$</td>
</tr>
<tr>
<td>$ak$</td>
<td>Action key of the device/user/intruder, $x \in {d, u, i}$</td>
</tr>
<tr>
<td>$cer$</td>
<td>Certificate of the device’s name/public key, $x \in {n, k}$</td>
</tr>
</tbody>
</table>

Before publishing data, the device must finish device discovery and naming service phases. Fig. 2 shows the actions.

• **a1:** A device sends an encrypted discovery request along with $cer_k$ to the aggregator.

• **a2:** When receiving the request, the aggregator decrypts it using $pu_k$ acquired from $cer_k$, and then verifies the device’s identity. If the device is legal, the aggregator sends $ak_d$ encrypted with $pr_k$ to the device.

• **a3:** The device obtains $ak_d$ through decryption, and then requests a name from the aggregator.

• **a4:** The aggregator sends a name request to the LSG via a secure channel which can prevent intruders from obtaining the request.

• **a5:** The LSG sends $cer_n$ to the aggregator.

• **a6:** The aggregator encrypts $cer_n$ using $ak_d$, and then provides it to the device.

• **a7:** The device gets $cer_n$ through decryption, and then publishes the data encrypted with $ak_d$ to the aggregator. In order to improve the security, the device also sends $cer_n$ encrypted with $pr_k$ to the aggregator.

Before getting data, the user needs to pass user registration and service discovery phases. The actions are given in Fig. 3.

• **b1:** A user initiates a registration request to the server.

• **b2:** When the request is received, the server sends a temporary password to the user via a secure channel.

• **b3:** Once receiving the password, the user changes it first, and then sends an action key request to the server.

• **b4:** The server assigns $ak_u$ encrypted by $pr_k_u$ to the user.

• **b5:** The user decrypts the message to get $ak_u$ using $pu_k_u$, and then sends the server a service request encrypted with $ak_u$ and $pr_k_u$.

• **b6:** The server decrypts the service request through two layers of decryption to verify the user’s identity. If the user is honest, the server requests the corresponding data from the LSG. If not, the service request is rejected.

• **b7:** The LSG requests the data from the aggregator.

• **b8:** The aggregator provides the data to the LSG.

• **b9:** The LSG forwards the received data to the server.

• **b10:** The server sends a message to the user who can obtain the data by decrypting the message with $ak_u$.

### B. CSP

Communicating Sequential Processes (CSP) is a process algebra proposed by C. A. R. Hoare [12]. Here we briefly introduce part of the CSP syntax used in this paper.

\[ P, Q ::= \text{Skip} \mid a \to P \mid c ? x \to P \mid c ! v \to P \mid P ; Q \mid P | Q \mid P \parallel Q \mid P < b > Q \mid P[[a \leftarrow b]] \]

- **Skip** means that a process terminates successfully.
- **$a \to P$** indicates that a process performs action $a$ first, and then acts like process $P$.
- **$c ? x \to P$** represents that a process receives a message via channel $c$ and assigns the received message to $x$, and then behaves like process $P$.
- **$c ! v \to P$** denotes that message $v$ is sent through channel $c$, and then process $P$ is executed.
- **$P ; Q$** is the sequential execution of processes $P$ and $Q$.
- **$P | Q$** describes that processes $P$ and $Q$ run in parallel.
- **$P \parallel Q$** stands for the general choice of processes $P$ and $Q$, and the selection is made by the environment.
- **$P < b > Q$** shows that if the condition $b$ is true, process $P$ is executed, otherwise process $Q$ is executed.
- **$P[[a \leftarrow b]]$** means renaming action. Event $a$ in process $P$ is replaced by event $b$.
### III. MODELING

In this section, we focus on the formal modeling of the ICN-IoT middleware architecture.

#### A. Sets, Messages and Channels

Before we investigate the formal model, we introduce some preparatory notations including sets, messages and channels.

First, we describe the related sets in this model. **Entity** set denotes entities including devices, aggregators, LSGs, ICN-IoT servers and users. **Req** set involves the request messages of entities. **Key** set represents all the keys including public key set $PuK$, private key set $PrK$ and symmetric key set $SmK$. **Data** set contains the data published by devices. **Con** set means other message contents involving certificate set $Cer$, feedback message set $Ask$ and password set $Pwd$.

Besides, we define the encryption function $E$ and decryption function $D$ to model the messages:

$$E(k, m); D(k, E(k, m)); D(k^{-1}, E(k, m))$$

Function $E(k, m)$ means that we encrypt the message $m$ using $k$. $D(k, E(k, m))$ denotes that we use a symmetric key $k$ to decrypt the message which is encrypted by $k$. $D(k^{-1}, E(k, m))$ indicates that we use the corresponding decryption key $k^{-1}$ to decrypt the message encrypted by $k$.

Based on the sets and functions defined above, we abstract and classify the messages as follows:

$$MSG_{req} = \{msg_{req}, a.b, req, msg_{req} \circ a, b, E(k_1, req), cer, msg_{key}, a.b, E(k_2, E(k_3, req)) \mid a, b \in Entity, k_1, k_2, k_3 \in Key, req \in Req, cer \in Cer\}$$

$$MSG_{key} = \{msg_{key}, a.b, E(k_1, k_2) \mid a, b \in Entity, k_1 \in Prk, k_2 \in Smk\}$$

$$MSG_{con} = \{msg_{con}, a.b, p, msg_{con}, a.b, E(k, c) \mid a, b \in Entity, p, c \in Con, k \in Key\}$$

$$MSG_{data} = \{msg_{data}, a.b, E(k_1, d), msg_{data}, a.b, E(k_2, E(k_3, d)) \mid a, b \in Entity, d \in Data, k_1, k_2, k_3 \in Key, c \in Cer\}$$

$$MSG = MSG_{req} \cup MSG_{key} \cup MSG_{con} \cup MSG_{data}$$

$MSG_{req}$ represents the set of request messages. $MSG_{key}$ means the set of messages containing action keys encrypted by private keys. $MSG_{con}$ set involves messages containing name certificates, feedback messages and passwords. $MSG_{data}$ is composed of messages containing the data published by devices. $MSG$ consists of all the messages in the model.

Then we give the definitions of communication channels:

- **Channels between devices, aggregators, LSGs, ICN-IoT servers and users described by COM_PATH:**  
  $ComDA$, $ComAL$, $ComSL$, $ComUS$

- **Channels for intruders to intercept or fake the transmitted messages denoted by INTRUDER_PATH:**  
  $FakeAD$, $FakeDA$, $FakeSU$, $FakeUS$

The declaration of channels is shown as follows:

$$Channel \ COM_PATH, INTRUDER_PATH : MSG$$

**B. Overall Modeling**

In this section, we give the whole model of the ICN-IoT middleware architecture. $System_0$ represents the system consisting only of legal entities. In order to simulate the real environment, we consider behavior of intruders. $System$ denotes the system which introduces the attacks from intruders.

$$System_0 =_{df} Device \| Agg \| LSG \| Server \| User$$

$$System =_{df} System_0 \| INTRUDER_PATH \| Intruder$$

$Device$, $Agg$, $LSG$, $Server$ and $User$ are processes describing the behavior of devices, aggregators, LSGs, ICN-IoT servers and users respectively. Besides, $Intruder$ process represents the actions of intruders such as intercepting and faking the messages transmitted among legal entities. The channels between processes are shown in Fig. 4.

**C. Device Modeling**

We formalize the process $Device_0$ to describe the behavior of the device without intruders as below:

$$Device_0 =_{df} ComDA[msg_{req}, D.A.E(prk, req), cer_k] \rightarrow$$

$$ComDA[msg_{key}, A.D.E(prk, ak_d)] \rightarrow$$

$$ComDA[msg_{req}, D.A.reqName \rightarrow$$

$$ComDA[msg_{con}, A.D.(ak_d, cer_n)] \rightarrow$$

$$ComDA[msg_{data}, D.A.E(prk, cer_n), E(ak_d, d) \rightarrow ComDA[msg_{con}, A.D.suc] \rightarrow Device_0]$$

$$\langle D(ak_d, E(ak_d, cer_n)) \rangle \triangleright \langle \text{fail} \rightarrow Device_0 \rangle$$

$reqName$ denotes the device’s name request. $d$ means the collected data. First, the device initiates a discovery request $req$ signed with $prk$ to the aggregator and then receives an action key $ak_d$. Then, the device sends the request $reqName$ to the aggregator, and obtains the name certificate $cer_n$. After acquiring $ak_d$ and $cer_n$, the device publishes the data $d$ encrypted with $ak_d$ to the aggregator. In order to improve the confidentiality and integrity of the system, the message for publishing data also contains $cer_n$ encrypted with $prk$. The above actions correspond to $a1 - a7$ in Fig. 2.

Now we consider the attacks from intruders. The process $Device$ with intruders is formalized via renaming as follows:
Device =⇒ Device0[
  ComDA!{ComDA} ← ComDA!{ComDA},
  ComDA!{ComDA} ← FakeDA!{ComDA},
  ComDA?{ComDA} ← ComDA?{ComDA},
  ComDA?{ComDA} ← FakeAD!{ComDA}]
{ComDA} means the set of all communication over the channel ComDA. The first two lines mean that whenever Device0 transmits a message on the channel ComDA, Device can transmit the same message on channel FakeDA or ComDA. The same is true for the last two lines.

D. Aggregator Modeling

The formal model of the aggregator abbreviated as $Agg_0$ without intruders is shown below:

$Agg_0 =_{df} ComDA!msgreq.D,A,E(prk_d, req).cer_k →$

- ComDA!msgreq.D,A,E(prk_d, ak_d) →
- ComDA?msgreq.D,A.reqName →
- ComAL?msg req.L.A.reqName →
- ComDA?msg req.L.A.reqName →

$\langle (D(prk_d, E(ak_d, cer_n))) \land D(ak_d, d)) \rightarrow (\text{fail} \rightarrow Ago)$

When receiving the device’s discovery request, the aggregator verifies its identity using the signature signed by prk_d. If the device is legal, the aggregator allows the device to join the system and assigns ak_d to it. Once the collected data are received from the device, the aggregator first verifies the authenticity of the device using the name certificate cer_n signed by prk_d. If the device is honest, the aggregator sends a positive feedback message suc to the device. The actions on the channel ComDA mean $a1 - a3$ and $a6 - a7$ in Fig. 2.

When receiving the LSG’s data request, the aggregator sends the requested data to the LSG via the channel ComAL. The actions on the channel ComAL represent $a4 - a5$ in Fig. 2 and $b7 - b8$ in Fig. 3.

The model of $Agg$ with intruders can be drawn via renaming similar to the process $Device$, we omit the details here.

E. User Modeling

The model of $User_0$ without intruders is given as below:

$User_0 =_{df} ComUS?msg req.U,S.reqReg →$

- ComUS?msg req.U,S.req key →

$\langle (D(ak_u, E(ak_u, cer_n))) \land (fpw \rightarrow User) \rightarrow (\text{fail} \rightarrow User) \rangle$

modifyPwd is a function to change the password. First, the user sends a registration request reqReg to the server and receives a temporary password. After the user changes the password using modifyPwd, it sends an action key request reqKey to the server. When receiving ak_u, the user sends the service request reqS encrypted by ak_u and signed with prk_u to subscribe to interested services. Finally, the user obtains the data for subscribed services through decryption. These actions denote $b1 - b5$ and $b10$ in Fig. 3.

The model of $User$ with intruders can also be acquired by renaming, we leave out the details here.

F. Server Modeling

We give the model of process $Server_0$ to describe the behavior of the ICN-IoT server without intruders as follows:

$Server_0 =_{df} ComUS?msg req.U,S.reqReg →$

- ComUS?msg req.U,S.req key →

$\langle (D(ak_u, E(ak_u, cer_n))) \land (fpw \rightarrow User) \rightarrow (\text{fail} \rightarrow Server) \rangle$

pwd is the temporary password sent to the user. req_data is the data request sent to the LSG. When the server receives the encrypted service request reqS, it checks if the user is legal using the signature signed by prk_u. If the user is legal, the server allows its service request, and then requests the corresponding data from the LSG. Otherwise, the service request is rejected by the server. After receiving the message containing requested data, the server first decrypts the message, and then sends the data encrypted with ak_u to the user. The actions on channel ComUS correspond to $b1 - b5$ and $b10$ in Fig. 3. The actions on channel ComSL denote $b6$ and $b9$ in Fig. 3. The model of Server considering intruders can be formalized via renaming as well, the details are omitted here. Similarly, we can define the model of process LSG.

G. Intruder Modeling

In order to better simulate the ICN-IoT middleware architecture in the real environment, we model the Intruder process which can intercept and fake messages among honest entities.

Firstly, we define the set of facts that the intruder can learn.

$\text{Fact} =_{df} \text{Entity} \cup \text{Puk} \cup \text{Cer} \cup \{E(k, d) | k \in \text{Key},$ $d \in \text{Data}\} \cup \text{MSG} \cup \{\text{Puk}_u, \text{prk}_u\}$

Through the known facts, the intruder can deduce new facts. The symbol $F \Rightarrow f$ means that the intruder can deduce a fact $f$ from the fact set $F$.

$\{k, c\} \Rightarrow E(k, c)$

$\{k^{-1}, E(k, c)\} \Rightarrow c, \{sk, E(sk, c)\} \Rightarrow c$

$F \Rightarrow f \land F \subseteq F' \Rightarrow F' \Rightarrow f$
The first rule means encryption. The second and third rules denote the decryption in asymmetric and symmetric encryption forms respectively. The last rule shows that if the fact \( f \) can be derived from a fact set \( F \), and \( F \) is a subset of \( F' \), then the intruder can also deduce \( f \) from the larger set \( F' \).

Moreover, we use a function \( \text{Info}(m) \) to imply the facts that the intruder can learn through intercepted messages.

\[
\text{Info}(\text{msg}_{req}.a.b.E(k_1, \text{req}).c) = \{ a, b, E(k_1, \text{req}), c \}
\]

\[
\text{Info}(\text{msg}_{key}.a.b.E(k_1, k_2)) = \{ a, b, E(k_1, k_2) \}
\]

\[
\text{Info}(\text{msg}_{con}.a.b.E(k_1, \text{con})) = \{ a, b, E(k_1, \text{con}) \}
\]

\[
\text{Info}(\text{msg}_{data}.a.b.E(k_1, d)) = \{ a, b, E(k_1, d) \}
\]

Besides, we introduce a channel \( \text{DEDUCE} \) for the intruder to deduce new facts. Its definition is given as below:

\[
\text{Channel} \ \text{DEDUCE} : \text{Fact}.P(\text{Fact})
\]

Then the process \( \text{Intruder}_0(F) \) can be modeled as follows:

\[
\text{Intruder}_0(F) = \{ \Box \in \text{MSG}_\text{Fake}.m \rightarrow \text{Intruder}_0(F \cup \text{Info}(m)) \}
\]

\[
= \{ \Box f \in \text{Fact}, f.P \rightarrow \text{Init}(k_l = \text{false}) \rightarrow \text{Deduce}.f.F \}
\]

\[
\rightarrow \{ (k_l = \text{true} \rightarrow \text{Intruder}_0(F \cup \{ f \})) \}
\]

When intercepting a message \( m \), the intruder adds \( \text{Info}(m) \) to its knowledge. If the intruder can decrypt \( m \), it can falsify \( m \) and send \( m \) to the original receiver. If the receiver does not recognize that the message has been modified, it means that the intruder successfully fakes as the original sender. Furthermore, the intruder can deduce new facts from its knowledge via the channel \( \text{DEDUCE} \) and add them to its knowledge. If the intruder successfully deduces action keys of the entities, action keys leakage occurs. Now we give the formal model of \( \text{Intruder} \), including its initial knowledge \( IK \).

\[
\text{Intruder} = \{ \Box \text{Intruder}_0(IK) \}
\]

where, \( IK = \{ \text{Entity} \cup \text{Puk} \cup \{ \text{prk}_1 \} \}

IV. VERIFICATION AND IMPROVEMENT

In this section, we analyze the verification results of the architecture. Based on the verification results and the analysis of attacks, we improve the original model and give the new verification results of the improved model.

A. Properties Verification

\[ \text{System}() \] means the model with intruders. We use Linear Temporal Logic (LTL) formulas to verify its properties.

**Property 1: Deadlock Freedom**

\[
\#\text{assert System}() \ \text{deadlockfree};
\]

The architecture should not run into a deadlock state. We verify this property by means of a primitive in PAT.

**Property 2: Data Availability**

\[
\#\text{define Data Availability data_success == true};
\]

\[
\#\text{assert System}() \ \text{reaches Data Availability};
\]

The property means that a legal user should get the required data. The assertion is used to check the property.

**Property 3: Action Keys Leakage**

\[
\#\text{define ActionKeys Leak Success kl == true};
\]

\[
\#\text{assert System}() \ | = ||! \text{ActionKeys Leak Success};
\]

As a vital part of the architecture, the leakage of action keys will cause a bad effect. We define a Boolean variable \( kl \) to check if the intruder can get the action keys, using the “always” operator \( [] \) in LTL.

**Property 4: Device Faking**

\[
\#\text{define Device Fake Success device_fake == true};
\]

\[
\#\text{assert System}() \ | = ||! \text{Device Fake Success};
\]

The property means that the intruder can pretend to be a legal device without being recognized. We adopt a Boolean variable \( \text{device_fake} \) for the verification in PAT.

**Property 5: User Faking**

\[
\#\text{define User Fake Success user_fake == true};
\]

\[
\#\text{assert System}() \ | = ||! \text{User Fake Success};
\]

The architecture should prevent intruders from subscribing to the services. If the intruder can fake as a legal user to obtain the data, many security issues may appear. We define a Boolean variable \( \text{user_fake} \) to verify the property.

Fig. 5: Verification results of the model

B. Verification Results

The verification results are shown in Fig. 5:

- **Property 1** is valid. It indicates that the proposed architecture will never get stuck in a deadlock situation.
- **Property 2** is valid, which shows that the data can be transmitted to the legal user who subscribes to the service.
- **Property 3** is invalid. It illustrates that the architecture can cause action keys leakage.
- **Property 4** is invalid. It means that the intruder can pretend to be a legal device to publish fake data.
- **Property 5** is valid, which represents that the intruder cannot disguise as a legal user successfully.

C. Attack Analysis

In this section, we discuss the reasons for the above insecure results. When the aggregator assigns \( ak_d \) to a device, it uses \( \text{prk}_4 \) to encrypt the message. Once getting \( \text{puk}_d \), the intruder can use it to decrypt the message to acquire \( ak_d \). Moreover, the intruder can tamper with the collected data and fake as the device to publish modified data. An example that leads to Device Faking and the leakage of \( ak_d \) is given as follows:
A1. \( D \rightarrow I : D.A.E(\text{prk}_d, \text{req}).\text{cer}_k \)
A2. \( I \rightarrow A : D.A.E(\text{prk}_d, \text{req}).\text{cer}_k \)
A3. \( A \rightarrow I : D.A.E(\text{prk}_d, \text{ak}_d) \)
A4. \( I \rightarrow D : D.A.E(\text{prk}_d, \text{ak}_d) \)
A5. \( D \rightarrow I : D.A.\text{reqName} \)
A6. \( I \rightarrow A : D.A.\text{reqName} \)
A7. \( A \rightarrow I : D.A.E(\text{ak}_d, \text{cer}_n) \)
A8. \( I \rightarrow D : D.A.E(\text{ak}_d, \text{cer}_n) \)
A9. \( D \rightarrow I : D.A.E(\text{prk}_d, \text{cer}_n).E(\text{ak}_d, d) \)
A10. \( I \rightarrow A : D.A.E(\text{prk}_d, \text{cer}_n).E(\text{ak}_d, \text{fakeD}) \)

- A1: The device sends a request to the aggregator.
- A2: The intruder eavesdrops on the request and gets \( \text{cer}_k \).
  Since the certificate is issued by CA and every entity has its public key, the intruder can decrypt \( \text{cer}_k \) to get \( \text{puk}_d \) using the public key of CA.
- A3: The aggregator uses \( \text{prk}_d \) to encrypt \( \text{ak}_d \) and sends the message to the device.
- A4: The intruder intercepts the message and decrypts it to get \( \text{ak}_d \) using the acquired \( \text{puk}_d \). At this point, the leakage of \( \text{ak}_d \) occurs.
- A5 – A8: The device requests for a name and receives \( \text{cer}_n \), during which the intruder intercepts the activities.
- A9: The device publishes collected data to the aggregator.
- A10: The intruder intercepts the message and gets the data using \( \text{ak}_d \). Then it disguises as the legal device to send modified data \( \text{fakeD} \) along with the original \( \text{cer}_n \) to the aggregator without being recognized.

Similarly, the intruder can get \( \text{ak}_u \) using \( \text{puk}_u \). Hence, the model cannot ensure the security of action keys and data.

**D. Improved Model and Verification**

In order to address the above issues, we improve the model by using a method similar to the digital signature. When distributing the action key, we use the receiver’s public key to encrypt the message. Furthermore, we introduce a method similar to digital signature for the aggregator to authenticate the device. That is to say, the device needs to sign with its private key when publishing the collected data. Therefore, the intruder can neither get the action keys nor fake as a legal device since it does not know \( \text{prk}_d \) or \( \text{prk}_u \). We modify the message definitions of the model. \( \text{MSG}_{\text{key}} \) and \( \text{MSG}_{\text{data}} \) are replaced by the following \( \text{MSG}_{\text{key1}} \) and \( \text{MSG}_{\text{data1}} \).

\[
\begin{align*}
\text{MSG}_{\text{key1}} &= \{ \text{msg}_{\text{key1}}.a.b.E(k_1, k_2) \mid a, b \in \text{Entity}, \ k_1 \in \text{Puk}, k_2 \in \text{Smk} \} \\
\text{MSG}_{\text{data1}} &= \{ \text{msg}_{\text{data1}}.a.b.E(k, E(k_1, d), c).E(k, d) \mid a, b \in \text{Entity}, k \in \text{Smk}, \ k_1 \in \text{Prk}, d \in \text{Data}, c \in \text{Cer} \}
\end{align*}
\]

Then we formalize the improved processes of \( \text{Device}_1, \text{Agg}_1, \text{LSG}_1, \text{Server}_1 \) and \( \text{User}_1 \) using the new message definitions. The improved model is given as follows:

\[
\begin{align*}
\text{System}_1 &=_{df} \text{Device}_1 || \text{Agg}_1 || \text{LSG}_1 || \text{Server}_1 || \text{User}_1 \\
\text{System} &=_{df} \text{System}_1 || \text{INTRUADER\_PATH} || \text{Intruder}
\end{align*}
\]

The verification results are shown in Fig. 6. **Property 3 and Property 4** are valid. It means that **Action Keys Leakage** and **Device Faking** problems are solved now.

**Fig. 6: Verification results of the improved model**

**V. CONCLUSION AND FUTURE WORK**

ICN-IoT middleware architecture is constructed by applying ICN into IoT. In this paper, we formalized the architecture using CSP. Feeding the model into PAT, we verified the functional and security properties of the model including deadlock freedom, data availability, action keys leakage, device faking and user faking. The verification results show that action keys leakage and device faking may occur once intruders appear. Thus, we improved the model by encrypting messages with the receiver’s public key. Moreover, we introduced a method similar to digital signature to the model. The new verification results indicate that the improved model can prevent intruders from invading the architecture. In the future, we will focus on more security issues of IoT systems. Formal methods will be used to verify other security properties of IoT systems.

**Acknowledgements.** This work was partly supported by National Key Research and Development Program of China (Grant No. 2018YFB2101300), National Natural Science Foundation of China (Grant No. 61872145, 62032024), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (Grant No. ZF1213).

**REFERENCES**


AnB2Murphi: A Translator for Converting Alice&Bob Specifications to Murphi

Yongxin Zhao¹, Hongjian Jiang¹, Jin Lv¹, Sijun Tan², Yongjian Li²∗

¹ Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
² State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract—As an important part of Internet of Things and 5G network technology, security protocols play a critical role in ensuring communication security. Formal analysis of security protocol has been successfully applied to find design flaws in recent years. Many formal verification tools have been used to verify the security protocols, including Murphi model checker. However, security protocols are often expressed in so-called Alice&Bob notation to describe the messages exchanged between honest principals, and security protocols defined by the A&B specifications cannot be applied to the formal verification tool directly. Therefore, there is a gap between Alice&Bob specifications and the modeling languages of the formal tools. In this paper, we propose AnB2Murphi, a novel and general translator which compiles the Alice&Bob specifications of security protocols into the input language of Murphi to bridge the gap. First, we specify the Alice&Bob specifications of the security protocol. Then we take the strand space as the intermediate form between A&B specifications and Murphi formal model. Finally, we use the Murphi model checker to verify the generated model of security protocol. The case studies of security protocols like Needham-Schroeder public key protocol and 5G EAP-TLS authentication protocol demonstrate the efficiency of our translator.

Index Terms—Murphi, Alice&Bob specifications, Strand Space, Security Protocol, Dolev-Yao

I. INTRODUCTION

As an important part of Internet of Things and 5G network technology, security protocols play a critical role in ensuring communication security. It simulates the communication process of multiple entities in the complex network environment and ensures the security of communication. When designing security protocols, such security properties should be ensured, including authentication, secrecy, et al., even with the presence of an intruder who can perform malicious actions. However, the design of these protocols is usually error-prone. This has led to the development of many verification theories and automatic verification tools such as ProVerif [1], Maude-NPA [2] and Murphi [3]. Formal methods have been used to verify security protocols for many years. However, it’s difficult for people who have no profound insights into verification theories to model the security protocols by these protocol analysis tools. Alice&Bob specifications for security protocols ranges from informal narrations of message flows to formal assertions of protocol properties. These message flows provide secure network communication by using the public key encryption.

In text-books, the Alice&Bob (abbr. A&B) notation [4] has been often used to describe the message exchanged between honest agents for the successful runs of security protocols. The following example expresses that Alice intends a message to Bob:

Alice → Bob : message

A&B notation is the most intuitive way to express the correspondence between principals of security protocols. However, the meaning of a protocol specification considered in a context only represents the ideal cryptosystem without active saboteurs. Moreover, A&B notation is too literary to be applied directly in formal verification or code implementation of security protocols. The modelling language of a formal verification tool is often low-level and detailed. Therefore, it is a cumbersome and time-consuming process to formally model security protocols.

The strand space model is a promising framework developed by Guttmann et al. to prove the correctness of security protocols. The strand in strand space represents a sequence of events which denote the execution of legitimate party in a security protocol or else a sequence of actions by a penetrator. A strand space is a collection of such strands. The graph structure is generated by causal interaction between strands. With the help of mathematically straightforward methods, strand space model justifies the correctness of security protocols.

In this work, we are motivated to implement a translator AnB2Murphi, which can convert A&B specifications of security protocols to Murphi automatically. We take strand space model as the intermediate form to ensure semantic consistency during the conversion processes. The existence of the intruders in the network and the attack ability of the intruders may pose a challenge to the security of the protocol. Therefore, we construct the deductive rules for intruders and model intruders’ behavioral capabilities based on Dolev-Yao model [5].

The main contributions of our work lies in the following aspects:

• **Automatic Translator.** We implement a translator AnB2Murphi to bridge the gap between high-level Alice&Bob specifications and low-level detailed Murphi model checker, which can convert the Alice&Bob specifications of security protocols to the input language of Murphi. Finally, AnB2Murphi has been successfully applied to several security protocols including typically 5G EAP-TLS authentication protocol. The verification result finds and reports the counterexample of errors in the design.
of the 5G authentication protocol, which demonstrates the efficiency of our translator.

- **Intruder Generation.** Based on Dolev-Yao model, we construct the deduction rules for active intruders, which can help simulate the possible attacks in an insecure network, such as replay attack, man-in-the-middle attack, etc. Besides, we have implemented the Diffie-Hellman exchange in our work which supports the algebraic operations and digital signatures to verify the TLS protocol.

The remainder of this paper is organized as follows. In Section II, we review the most related works. In Section III, we give a brief introduction of A&B specifications, strand space and Murphi model checker. In Section IV, we present the architecture of AnB2Murphi and the corresponding relationship between Alice&Bob specifications and Murphi model, then we elaborate the implementation details. In Section V, we report the verification results of the generated Murphi model. In Section VI, we conclude this paper and discuss the future work.

II. RELATED WORK

There has been a lot of discussion on formal verification of security protocols [6]. Besides, the research of A&B notation has received considerable attention. In [7], the authors generalized the formal protocol specification languages and gave the formal semantics for a language based on Alice and Bob including algebraic reasoning. But it was still not expressive enough. In [8], the authors proposed a formal protocol specification language based on the popular Alice&Bob notation, that was AnBx. This specification language extended the formal semantics of A&B notation with a novel notation of forwarding channels. This inspires me to use specific term tmp to represent the forward specific messages in the implementation process. In [9], Omar and Sebastian et al. formalized the language SPS and an automatic translation to robust real-world implementations and corresponding formal models, whose translation was effective.

Besides, there is also a lot of research work on converting A&B specifications to detailed implementation. David Basin et al. [10], they translated A&B protocol specifications to the input language of Tamarin. However, Tarmin [11] is a prover based on theorem proving, which verifies the correctness of the protocol by using multiset rewriting. It requires the user to understand the protocol and supply auxiliary lemmas by heuristics, which is hard even for experts. In contrast, the authors presented a methodology for using Murphi to analyze security protocol in [12]. Murphi is a model checker that supports verification with parameters and has been successfully applied to several industrial protocols. Based on the previous research work, we can see that it is significant to automatically generate the Murphi formal model from the A&B protocol specifications to verify these security protocols.

III. PRELIMINARIES

In this section, we present A&B specifications of security protocols and introduce strand space and Murphi.

A. Formal A&B Specifications Syntax

In cryptography, Alice&Bob specifications are commonly used to describe security protocols. The A&B specifications are intuitive, succinct and yet expressive. The security protocol is specified as a list of message exchange steps of the following form:

\[ A \rightarrow B : \text{message} \]

where the initiator A sends the message to the responder B.

As shown in Fig.1, we take Needham-Schroeder public key protocol [13] as the example. The A&B specifications consist of the following parts:

- **Types.** This part declares all identifiers of the protocol specification. In the example, we specify agents A and B, PK is a function that we yield the public key for agents.

- **Knowledge.** This part specifies the initial knowledge attached to each regular agent, consisting of a set of messages. We define the term \( \text{initKnw} \text{Role}(\text{Role}) \) to be the initial knowledge of the agent Role. For instance, \( \text{initKnw} \text{Role}(A) = \{A, B, Na\} \) in the example.

- **Agents.** Agents is the core of A& specifications which describes the ideal, safe run of the protocol. Every Agent contains two parts: the agent’s knowledge and the list of actions to exchange messages. Actions defines the process of protocol execution.

- **Environments.** This part defines a protocol instance, in which we need to give the actual parameters to instantiate the formal atomic parameters in the Knowledge part. See that \( \text{Init}[1] \) is an instance of agent A and the initial knowledge of A[1] is \(< Alice, Intruder, Na >\).

- **Goals.** We specify the security properties that the protocol should achieve. In this work, we mainly focus on goals such as secrecy and non-injective agreement.

<table>
<thead>
<tr>
<th>Protocol: Needham-Schroeder public key</th>
<th>Types:(+ Global Types+)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agent: A, B;</td>
<td>Function: PK;</td>
</tr>
<tr>
<td>Knowledge:(+ Initial Knowledge+) A: A, B, Na</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B: B, Nb</td>
</tr>
<tr>
<td>Agents:</td>
<td></td>
</tr>
<tr>
<td>Init(A,B,Na)</td>
<td>[1]+, B, (Na, A, B): Na, A\text{PK}(B)</td>
</tr>
<tr>
<td></td>
<td>[2] - Na, Nb, PK(A)</td>
</tr>
<tr>
<td></td>
<td>[3]+, B, (): Nb\text{PK}(B)</td>
</tr>
<tr>
<td>Resp(B,Nb)</td>
<td>[1] - Na, A\text{PK}(B)</td>
</tr>
<tr>
<td></td>
<td>[3] - Nb\text{PK}(B)</td>
</tr>
<tr>
<td>Environments:</td>
<td>(Protocol Instance+)</td>
</tr>
<tr>
<td>[agents1]Init[1]: &lt; Alice, Intruder, Na &gt;</td>
<td></td>
</tr>
<tr>
<td>[agents2]Resp[1]: &lt; Bob, Nb &gt;</td>
<td></td>
</tr>
<tr>
<td>Goals: (+Security Goals+)</td>
<td></td>
</tr>
<tr>
<td>[secrecy] Nb secret of &lt; A, B &gt;</td>
<td></td>
</tr>
<tr>
<td>[weakB] B non – injectively agrees with A on Na</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1: The Needham-Schroeder Public Key Protocol in A&B Specifications
\section*{B. Extended Strand Space}

The A&B specifications is a relatively high-level language without implementation details, which cannot be directly used for verification with model checking tools. The strand space \cite{14}, which is one of the most successful and widely used formalizations, serves as the intermediate representation format of A&B specifications.

The security protocol defines a sequence of message-exchanged events for each agent. In strand space theory, an action that agents can take during the execution of security protocols includes sending a message and receiving a message. We denote the sending and receiving action by a set of two signs $\text{Sign} = \{+, -\}$, respectively. An event is a pair $(\sigma, m)$, where $\sigma \in \text{Sign}$ and $m$ is a message. For example, a node in a strand is like $(+, m)$ which means that the owner of the strand sends a message $m$.

A strand represents a sequence of events of an agent in a particular protocol run. A strand element is called a node. If $s$ is a strand, $(s, i)$ denotes $i^{th}$ node in strand $s$. In NSPK protocol, the strand of agent $A$ specifies a sequence of events which can be seen on the left of Fig.2. In this strand, the agent $A$ sends a message $\langle [Na, A] \rangle_{PK(B)}$ to agent $B$, and expects to receive back a message of the form $\langle [Na, Nb] \rangle_{PK(A)}$, after which it will send $\langle Nb \rangle_{PK(B)}$.

A strand space $\Sigma$ is a set of strands of the principals participating in the running protocol. We have drawn the strand space model of NSPK protocol as shown in Fig. 2, from which we can see the following relationships:

- The relation $n \rightarrow n'$ represents the inner-strand communication. It holds between nodes $n$ and $n'$ if $n = (s, i)$ and $n' = (s, i + 1)$.
- The relation $n \rightarrow n'$ represents the inter-strand communication. The inter-strand communication $(s_B, 1) \rightarrow (s_A, 1)$ represents that agent $A$ is sending a message $\langle [Na, A] \rangle_{PK(B)}$ out, and agent $B$ will finally receive a message like this.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig2.png}
\caption{Regular Strands of NSPK Protocol}
\end{figure}

\section*{C. Murphi}

Murphi is an enumerative (explicit state) model checker with its own input language, which supports the guard $\rightarrow$ action notation. It can abstract the behavior of a system and simulate the running rules of this system. Murphi model checker has a formal verifier that is based on explicit state enumeration, which has been successfully applied to several industrial protocols. States encountered in this mode are saved in a global hash table. States generated that exist in the hash table are not expanded. The description of Murphi input language consists of declaration part, transition rule part, initialization part and property part.

\begin{verbatim}
Declaration Part  
--Constant declarations 
--Type declarations 
--Global variable declarations 
--Procedure and function declarations

TransitionRulePart
rule "ruleName"
  guard part -- conjunction of predicates  
  action part -- a set of statements
endrule

InitializationPart
startstates
  --initial the value of variables
end

PropertyPart
  invariant "inv"
  -- define the security property 
end
\end{verbatim}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{The Basic Structure of Murphi Model}
\end{figure}

The basic structure of a Murphi model can be seen in Fig.3, among which the most important is the transition rule part to describe the transition from one state to another state. A transition rule mainly consists of two parts: guard and action. Only if the predicates in the guard are satisfiable can the statements in the action part execute. Given a Murphi model, the model checker Murphi will enumerate the entire state space explicitly until no new reachable state can be explored or the properties fail to hold on to the protocol. Relatively, a set of all possible reachable states is regarded as the reachable state set (abbr. $RS(P)$). Murphi starts from the initial state, which is prescribed in the initialization part. Then, it will randomly choose a transition rule whose guard is satisfied and execute the corresponding action.

\section*{IV. Translate to Murphi}

In this section, we discuss how to translate A&B specifications to Murphi. Meanwhile, we take NSPK protocol as a running example.

\subsection*{A. The Architecture of AnB2Murphi}

First of all, The overall architecture of AnB2Murphi shown in Fig.4 consists of two main phases: Converting and Verification.
-- Converting. This phase aims at converting high-level A&B specifications of security protocols into corresponding Murphi model. It includes two important processes: parsing and generating. The parsing process uses Ocaml/menhir tools to analyze and transform the structure of A&B specifications into the extended strand space. The generating process is the focus of our framework. After parsing the A&B specifications, we generate the transition rule part and the built-in environment into the Murphi model based on the extended strand space.

- Verification. This phase aims at checking the Murphi model generated in the converting phase. Murphi compiler first compiles the model into a C++ file, then uses the compiling and executing mode of the C++ program to perform the verification process, which accelerates the speed of model detection.

We have given the description of A&B specifications and the structure of Murphi model in Section.III. The corresponding relationship between the A&B specifications and Murphi is shown in Fig.5. The rest of this section focuses on the implementation phases, mainly including Converting and Verification.

B. Converting

Converting is mainly constructed by Parsing and Generating processes. The Parsing process transforms the A&B specifications into the extended strand space to ensure the consistency of semantic during the conversion processes. And the Generating process is used to generate transition rule part and built-in environment in Murphi model.

1) Parsing: The main task of this process is to analyze the structure of A&B specifications by tool Ocaml/Menhir. This process is relatively simple and we will not elaborate on it too much. The result of parsing A&B specifications is given directly below.

type ProtocolContext = [ ]
| Protocol of type * knowledge * agent * environment * goal |
| Null |

We translate the security protocol as a type of ProtocolContext, which includes the following five parts: type, knowledge, agent, environment and goal. The Protocol is the tag of a protocol. Therefore, the A&B specifications of a protocol will be parsed into Protocol(t, k, a, e, g), in which the element in the quintuple are corresponding to the five parts of the A&B specifications, respectively.

In addition, the possible messages which can be exchanged between agents in security protocols are defined below:

```
| Type message = |
| Nonce of identifier (*Nonce*) |
| Agent of roleName (*Agent Identifier*) |
| Const of identisirt (*Constant Number*) |
| Pk of roleName (*Public Key*) |
| Sk of roleName (*Secret Key*) |
| K of roleName * roleName (*Symmetry Key*) |
| Mod of message * message (*Mod*) |
| Exp of message * message (*Exp*) |
| Tmp of messageName (*Temporary*) |
| Sign of message * message (*Signature*) |
| Aenc of message * message (*Aencrypt*) |
| Senc of message * message (*Sencrypt*) |
| Concat of message list (*Concatenation*) |
```

Fig. 5: Correspondence of A&B Specifications and Murphi

The message consists of atomic messages and compound messages. Atomic messages are non-divisible messages, such as Nonce, Agent, Pk and so on. Compound messages are composed of multiple atomic messages, which are encrypted symmetrically or asymmetrically, or combined by connections.

According to the Agents part of in A&B specifications, we construct the extend strand space model of the protocol by using the following methods.

(1) genStrand(agent, rolei): This method converts the Agents in A&B specifications into a node of strand rolei,

genStrand(Agent(roleName, message list, action list), rolei) = action list if roleName = rolei.

In the A&B specifications, Agents is expressed into the form of Agent(roleName, message list, action list). If rolei is the name of the agent, then the result of method genStrand is the corresponding action list.
The intruder in the Murphi model can deduce the message according to the deduction rule listed in Table I. Besides, it keeps the \( K_{\text{Knws}} \) which stores the initial and deductive knowledge the intruder holds. As for the example, the public keys of agents \( A \) and \( B \) are in the intruder’s knowledge set initially. After intercepting a message from the network, the intruder makes the deduction of the message based on the knowledge, and adds the knowledge into his knowledge set \( K_{\text{Knws}} \).

<table>
<thead>
<tr>
<th>Decryption</th>
<th>(decrypt) ( {m}<em>K \in K</em>{\text{Knws}} \land \text{inv}(K) \in K_{\text{Knws}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encryption</td>
<td>(encrypt) ( {m}<em>K \in K</em>{\text{Knws}} )</td>
</tr>
<tr>
<td>Separation</td>
<td>(deconcat) ( {m_1}<em>K \in K</em>{\text{Knws}} \land {m_2}<em>K \in K</em>{\text{Knws}} )</td>
</tr>
<tr>
<td>Concatenation</td>
<td>(enconcat) ( {m}<em>K \in K</em>{\text{Knws}} \land {m}<em>K \in K</em>{\text{Knws}} )</td>
</tr>
<tr>
<td>Signature</td>
<td>(signature) ( {m}<em>{SK} \in K</em>{\text{Knws}} )</td>
</tr>
<tr>
<td>Verify</td>
<td>(verify) ( {m}<em>{SK} \in K</em>{\text{Knws}} )</td>
</tr>
<tr>
<td>Hash</td>
<td>(hash) ( m \in K_{\text{Knws}} )</td>
</tr>
</tbody>
</table>

After generating transition rules which include the transition rules for regular principals and the potential intruders, the built-in environment is then generated to define the procedures and functions in Murphi. It mainly includes the construction procedure and destruction procedure of the exchanged messages.

(1) Extracting the exchanged messages from actions and generating the \( \text{patlist} \) consisting of the message patterns and its sub-patterns:

\[ \text{patList} = \text{rmEquiv}(\text{delDup}(\text{getPatList}(\text{actions}))) \]

There are three methods involved in the derivation of this process in Ocaml. Method getPatList(actions) extracts the messages and their sub-messages from actions and returns a pattern list. Then method delDup(list) is used to delete the duplicate items in the list and return without replicas. Finally, method rmEquiv(list) helps remove the equivalence class in the list. For example, \( \text{Agent}(A) \) and \( \text{Agent}(B) \) belong to the same class and just keep one.

(2) Generating the construction procedures of messages. Based on the generated pattern list, we generate the corresponding built-in construction and destruction procedure for each pattern. In Murphi, we use a global array \( \text{msgs} \) to store message generated during the protocol execution. During construction process, we first find out whether the message exists in the \( \text{msgs} \). If it exists, we directly returns its index; if not, we construct the message and add the index of message into \( \text{msgs} \).

C. Verification

In Converting phase, we generate the Murphi model of the security protocol illustrated in A&B specifications, and we
output the results to the file result.m.

Murphi model checker will first check whether there exists some errors in the Murphi model. If not, Murphi compiler convert the result.m into a C++ program result.cpp. Then, we compile the C++ program using GNU compiler g++, which gives us the resulting executable, a verifier that computes reachability for the specific problem. Finally, execute the program result.out to verify the security protocols, and the result of the verification will be output to the terminal.

V. Evaluation

We have implemented several security protocols in A&B specifications which come from the security protocols repository [15]. Significantly, we model the 5G EAP-TLS Authentication protocol which plays a critical role of the first safeguard in ensuring the communication security. There are up to 17 messages-exchanged in the authentication protocol which involves sophisticated cryptography terms. When verifying this complicated model, Murphi needs to explore the whole state space. Basing on cmurphi5.4.9.1 as the verification engine, we find that the weak-agreement and secrecy prekey are violated. Murphi generates counterexamples for these two properties.

The experiments are carried out on a PC equipped with the macOS Catalina and Intel Core i7 with 2.6Ghz CPU and 16GB RAM. The verification result of these security protocols are shown in Table II. The source code of translator AnB2Murphi can be accessed at [16].

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Unsatisfied</th>
<th>Time (sec.)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeedhamSchroder</td>
<td>secrecy(Nb)</td>
<td>0.10</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>weakB</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>Lowe’s NeedhamSchroder</td>
<td>no error</td>
<td>0.10</td>
<td>58</td>
</tr>
<tr>
<td>Diffie–Hellman</td>
<td>secrecy(Na)</td>
<td>0.10</td>
<td>64</td>
</tr>
<tr>
<td>Otway–Rees</td>
<td>no error</td>
<td>2.13</td>
<td>117</td>
</tr>
<tr>
<td>CCITT X.509(1)</td>
<td>secrecy(Ya)</td>
<td>0.21</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>weakB</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td></td>
<td>weakA</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>CCITT X.509(1c)</td>
<td>no error</td>
<td>0.45</td>
<td>69</td>
</tr>
<tr>
<td>Woo and Lam Pi</td>
<td>secrecy(Nb)</td>
<td>0.10</td>
<td>69</td>
</tr>
<tr>
<td>Andrew Secure RPC</td>
<td>secrecy(Kab)</td>
<td>2.77</td>
<td>54</td>
</tr>
<tr>
<td>EAPTLS authentication</td>
<td>secrecy(prekey)</td>
<td>1.21</td>
<td>1700</td>
</tr>
<tr>
<td></td>
<td>weakC</td>
<td>151.55</td>
<td></td>
</tr>
</tbody>
</table>

VI. Conclusion and Future Work

In this paper, we have implemented an automatic translator AnB2Murphi to bridge the gap between high-level Alice&Bob specifications and low-level Murphi model checker, which can help verify the security protocol described in the A&B specifications. We design a scheme to translate the actions of regular principals into strand space, which can well describe the communication relationship between principals and the state transition of themselves. Based on Dolev-Yao model, we construct the deduction rules for intruders, which can help simulate the possible attacks in insecure networks. The translator is implemented in Ocaml/Menhir, which is a simple but powerful parser generator for the Ocaml programming language. AnB2Murphi has been successfully applied to several typical security protocols. The results of verification are consistent with those already proved.

For the weakness of the current work, we would like to point out that the A&B specifications of a protocol are intuitive. Because Murphi is a model checker, it is good at examine multiple runs of protocol, and give us the counterexample trace when the protocol does not satisfy the specification. But it bothers us to specify the actual parameters in the environment of A&B specifications. We will fix this question by integrating machine learning into the work.

Acknowledgements

This work is supported by National Key Research and Development Program (2020AAA0107800), Shanghai Science and Technology Commission Program under Grant 20511106002, Grant 61672503 from National Science Foundation in China.

References

Formal verification of multitask hybrid systems by the OTS/CafeOBJ method

Masaki Nakamura 1 Kazutoshi Sakakibara 1 Yuki Okura 1 Kazuhiro Ogata 2
1Toyama Prefectural University, Toyama, Japan 2Japan Advanced Institute of Science and Technology, Ishikawa, Japan

Abstract- Hybrid systems combine both continuous and discrete behaviors. Formal descriptions of hybrid systems may help us to verify desired properties of a given system formally with computer supports. In this paper, we propose a way to describe a formal specification of a given multitask hybrid system as an observational transition system in CafeOBJ algebraic specification language and verify it by the proof score method based on equational reasoning implemented in CafeOBJ interpreter.

Keywords-component: hybrid system; algebraic specification; observational transition system; proof score method

II. Preliminaries

A Hybrid automata of a signal control system

In this article, we consider a hybrid automaton of a simple signal control system, represented in Fig.1. The system consists of a signal and a car such that the car is prohibited from being in a specific area, between \( c_{s0} \) and \( c_{s1} \), while the signal is red. The specific area is called the critical section. The signal has three modes indicating its color label. Each color of the signal should be kept more than \( t_0 \) time units. The car has two modes: going and not-going. In the going mode, the car moves forward according to time advancing. The car stays there in the not-going mode. If the signal label is not green, the car cannot enter the critical section. If the signal label is changed into yellow while the car exists in the critical section, the car should keep the going mode, that is, it should not stop. Thus, if the interval \( t_0 \) is more than the time which the car needs to go through the critical section \( (c_{s1} - c_{s0}) \), the car does not exist in the critical section while the signal is red.

Hybrid automata are models of hybrid systems with discrete and continuous behavior. We give a model of our simple hybrid system according to the literature [5]. Figure 2 represents a hybrid automaton

\( (L_{oc}, Lab, Edg, X, Init, Inv, Flow, Jump) \)

of our simple signal control system with a single car. Locations \( L_{oc} \) and edges \( Edg \) with labels \( Lab \) are depicted as circles and arrows between them. \( X \) is given as \( \{pos, now, l\} \). The initial values are all zero, that is, \( Init(pos) = Init(now) = Init(l) = 0 \), where \( pos \) and \( now \) stand for the position of the car and the current time, and \( l \) is...
used for the signal interval control. \( \text{Loc} = \{g, y, r\} \times [0, 1] \times [0, 1] \) is the set of locations where \((\text{label}, \text{going}, \text{cs}) \in \text{Loc}\) represents the location where the color of the signal is \text{label}, the car moves when \text{going} and the car exists in the critical section when \text{cs}. The top, the middle and the bottom labels in a location correspond to locations, flow conditions and invariants of the location respectively. For example, \( \text{Flow}(l) = \text{STOP} \) and \( \text{Inv}(l) = \text{CS} \) for \( l = (g, 0, 0) \).

The flow conditions of locations are given by \( \text{STOP} : \text{pos}' = 0 \land \text{now}' = 1 \) or \( \text{GOING} : \text{pos}' = 1 \land \text{now}' = 1 \), where \( x' \) stands for the time derivative of \( x \). Since \( \text{now}' = 1 \) holds in each location, the value of \( \text{now} \) always increases according to time advancing. Thus, \( \text{now} \) keeps the elapsed time from the initial state. The value of \( \text{pos} \) is unchanged when \( \text{STOP} \) holds, and increases when \( \text{GOING} \) holds. The invariants \( \text{CS} : \text{cs}_0 \leq \text{pos} \leq \text{cs}_1 \) and \( \overline{\text{CS}} : (0 \leq \text{pos} \leq \text{cs}_0) \lor (\text{cs}_1 \leq \text{pos} \leq \text{cs}_1) \land \text{cs}_1 \leq \text{pos} \) mean that the car exists and does not exist in the critical section respectively.

The edges \( e \) between different color’s locations are labeled by \text{change} and the jump condition \( \text{Jump}(e) \) is given as \( \text{CH} : l \leq \text{now} \land l^* = \text{now} + t_0 \), which means that the edge can be executed when \( l \leq \text{now} \) and then the value of \( l \) is updated by \( \text{now} + t_0 \). We omit some of \text{change} and \text{CH} in Figure 2. We also omit unchanged variables in the jump condition, that is, \( x'^r = x \) for each variable \( x \).

The state transition system is obtained by a hybrid automaton, where a state \((l, v) \in S\) is a pair of a location and values of variables. From the above conditions, there is no path \( s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n \) such that \( s_0 \) is an initial state, and \( s_n = (l, r, x, y, v) \) and \( v(\text{pos}) < v(\text{cs}_1) \), that is, the car does not exist in the critical section when the signal is red. We call it the safety property.

Hybrid automata of plural cars can be obtained by composing the copies of the above hybrid automaton. Consider the hybrid automaton with two cars. A location is a pair \((l_1, l_2)\) where \( l_i = (\text{label}_i, \text{going}_i, \text{cs}_i) \). There is an edge between \((l_1, l_2)\) and \((l'_1, l'_2)\) if there is an edge \( e \) between \( l_i \) and \( l'_i \) and \( l_j = l'_j \) \((i, j \in \{1, 2\}, i \neq j\)\). When the number of cars increases, the state space is exponentially increases, e.g., the number of locations becomes \( 10^n \) for \( n \) cars.

In the following sections, we describe OTS/CafeOBJ specifications of the above hybrid automaton with a single car and with plural cars, and give a formal proof of the safety property.

\section{The OTS/CafeOBJ method}

We introduce some notion and notation of the OTS/CafeOBJ method including the proof score method [1].

A CafeOBJ specification consists of modules. A CafeOBJ module \( \text{SP} = (\Sigma, \text{ES}) \) consists of its signature and axioms. A signature \( \Sigma = (\Sigma_0, \Sigma_1, \text{ES}) \) consists of a set of sorts, an ordering on the sorts and an \( \text{ES}-\)sorted set of operations. A \( \Sigma \)-algebra \( A \) is an algebra which has a carrier set \( A_s \) for each \( s \in \Sigma \) and an operation \( A_f : A_{s_0} \times \cdots \times A_{s_n} \rightarrow A_s \) for each \( f \in (\text{ES})_{s_0\ldots s_n} \). An axiom \( l = r \) if \( c \in E \) is a conditional equation whose both sides \( l \) and \( r \) of the equation are terms of a same sort and condition is a term of boolean sort constructed from the operations in \( \Sigma \) and variables. A \( \Sigma \)-algebra is an \( \Sigma \)-algebra which satisfies all equations in \( \text{ES} \). When SP has tight denotation, it denotes the initial SP-algebra. When SP has loose denotation, it denotes all \( \Sigma \)-algebras.

The following is a loose module specifying an arbitrary set with a binary predicate.

\begin{verbatim}
mod* PID { [ [PId] ]
op _-_ : Pid Pid -> Bool {comm} }
\end{verbatim}

A loose module (mod*) denotes all models satisfying axioms. Module PID denotes an arbitrary set with a binary relation. The following is a tight module specifying labels of a traffic signal.

\begin{verbatim}
mod! LABEL { [Label]
ops gr re ye : -> Label
pred _-_ : Label Label {comm}
op next : Label -> Label
eq (L:Label = L) = true .
eq (gr = re) = false .
eq (gr = ye) = false .
eq (re = ye) = false .
eq next(re) = gr .
eq next(gr) = ye .
eq next(ye) = re .
}
\end{verbatim}

A tight module (mod!) denotes the initial model. In the initial mode, any elements of a carrier set is represented by a term constructed from its signature, and no two elements of a carrier set are equivalent unless the corresponding terms can be shown to be equal using its axioms. Module LABEL has three constant operators \text{gr}, \text{ye} and \text{re} of Label, a binary predicate \text{=_} and a unary operation \text{next} on Label. The first four equations define the equality predicate, which
takes two labels and returns true if they are same, otherwise false. The unary operation next returns a next label, defined by re ⇒ gr ⇒ ye ⇒ re.

An OTS/CafeOBJ specification consists of data modules and a system module. Modules LOC and PID are examples of data modules. A system module is given as a behavioral specification of CafeOBJ. A behavioral specification has a special sort, called a hidden sort, and special operations, called behavioral operations, whose arguments include the hidden sort. A behavioral operation whose returned sort is not hidden is called an observation. A behavioral operation whose returned sort is hidden is called a transition. Two elements of the hidden sort are observationally equivalent if their observed values are equivalent for each observation. An OTS/CafeOBJ specification is a restricted behavioral specification, where observational equivalence is preserved by transitions.

The following is an OTS/CafeOBJ specification of a signal control system:

```plaintext
mod* SIGNAL{ pr(LABEL) *[Sys]*
op init : -> Sys  bop color : Sys -> Label
bop change : Sys -> Sys
eq color(init) = gr .
eq color(change(S:Sys)) = next(color(S)) . }
```

Module SIGNAL imports module LABEL with the protecting mode, where a model of the importing module includes a model of the imported module as it is. Hidden sort Sys is declared, which denotes the state space of a system to be specified. Constant init is declared as an initial state. Observation color observes a color, where term color(s) represents the current color in state s of the signal control system. Transition change changes a color, where term change(s) represents the state obtained after changing.

The first equation color(init) = gr specifies the initial state through observation such that the initial color is green. The second equation specifies the behavior of transition change through observation. Term change(S) represents the state obtained by applying change to S. The color of change(S) is defined as the next color of S. For example, color(change(change(init)))) is equivalent to re.

III. OTS/CafeOBJ SPECIFICATIONS OF HYBRID SYSTEMS

In this section, we introduce a way to describe OTS/CafeOBJ specifications of hybrid systems. We model signal control systems with a single car and with plural cars as hybrid automata, and describe them as OTS/CafeOBJ specifications.

A An OTS/CafeOBJ specification of a signal control system with a single car

In our OTS model of the signal control system, there are three observations for discrete variables and three observations for continuous variables. Observations color, going and cs observe the value of elements of a location. Observations now, pos and l observes the value of those variables. There are two kinds of transitions in OTS models: discrete and continuous transitions. Discrete transitions go, stop, in, out and change correspond to edges. A continuous transition tick, advances the system’s time by $t \in \mathbb{Q}^+$.

Table 1 shows the correspondence between the hybrid automaton in Section 1 and our OTS/CafeOBJ specification.

We give a system module SIGNAL which imports the built-in module RAT of rational numbers and LABEL given in the previous section. The following is a part of module SIGNAL specifying the initial state.

```
| eq now(init) = 0 . | eq pos(init) = 0 . |
| eq going(init) = false . | eq cs(init) = false . |
| eq color(init) = gr . | eq l(init) = 0 . |
```

The initial values of now, pos and l are defined as 0. The initial values of going and cs are false. The initial color is green. Module SIGNAL includes declaration ops cs0 cs1 t0 : -> Rat of constants cs0, cs1 and t0.

Transition change is specified as follows:

```
| eq c-change(S) = 1(S) <= now(S) . |
| ceq change(S) = S if not c-change(S) . |
| ceq color(change(S)) = next(color(S)) if c-change(S) . |
| ceq l(change(S)) = now(S) + t0 if c-change(S) . |
| eq now(change(S)) = now(S) . |
```

The effective condition c-change(S) is defined in the first equation such that $l \leq now$. The second (conditional) equation specifies that the state is unchanged if change is not effective. The third and fourth equations specify the updated values of color and l such that color becomes the next color.

**Table 1. The correspondence between our hybrid automaton and OTS/CafeOBJ specification**

<table>
<thead>
<tr>
<th>Hybrid Automaton</th>
<th>OTS/CafeOBJ specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loc</td>
<td>color, going, cs</td>
</tr>
<tr>
<td>Edg</td>
<td>go, stop, in, out, change</td>
</tr>
<tr>
<td>X</td>
<td>now, pos, l</td>
</tr>
<tr>
<td>Init</td>
<td>init</td>
</tr>
<tr>
<td>Inv</td>
<td>c-tick</td>
</tr>
<tr>
<td>Flow</td>
<td>observed values after tick</td>
</tr>
<tr>
<td>Jump</td>
<td>effective conditions and observed values after transitions</td>
</tr>
</tbody>
</table>

*We assume a temporal domain is dense, that is, there is a point between any pair of points. We use rational numbers for the clock values.*
and $l$ is set to $t_0$ time later. The remaining equations specify other variables are unchanged.

Transition $in$ is specified as follows:

\[
\begin{align*}
\text{eq } c\text{-in}(S) &= (c_0 = \text{pos}(S) \text{ and color}(S) = \text{gr}) . \\
\text{eq } cs\text{(in}(S)) &= \text{true if } c\text{-in}(S).
\end{align*}
\]

Transition $in$ is effective when the car exists at $c_0$ and the signal is green. When $in$ is effective, $cs\text{(in}(S))$ becomes true, that is, location $(\_\_\_, 0)$ is changed into location $(\_\_\_, 1)$. The other discrete transitions are defined similarly.

Next, we specify the continuous transition. Time advancing $tick$, is described as follows:

\[
\begin{align*}
\text{eq } now\text{(tick}(X, S)) &= now(S) + X \text{ if } c\text{-tick}(X, S) . \\
\text{eq } pos\text{(tick}(X, S)) &= \text{(if going(S) then pos(S) + X else pos(S)) fi when } c\text{-tick}(X, S).
\end{align*}
\]

Term $tick(r, s)$ is the result state of applying $tick$, to state $s$. Since $now' = 1$, the value of $now$ increases by $1 \times r$. If we have more complex differential equations in $Flow$, equations become larger.

For a given $r \in Q$, when $tick$, is done, the current time $now$ increases by $r$. The position $pos$ increases by $r$ if going is true, that is, the car moves forward. The effective condition $c\text{-tick}$ is given by invariants of the hybrid automaton.

\[
\begin{align*}
\text{eq } c\text{-tick}(X, S) &= 0 <= X \text{ and } X <= cs_1 - cs_0 \text{ and } \\
&\text{cs_0 <= pos}(S) + X \text{ and } pos(S) + X <= cs_1 \text{ and } \\
&\text{cs_1 < pos}(S) + X \text{ implies not cs_0(S)).}
\end{align*}
\]

Invariants in hybrid automata should always hold, which means that time cannot advance if the invariants do not hold. Thus, the effective condition $c\text{-tick}$ is given as the conjunction of all invariants.

**B An OTS/CafeOBJ specification of a signal control system with plural cars**

Consider the case that more than one cars appear in our signal control system. The system is an example of multitask hybrid systems. In OTS models of multitask systems, observations and transitions related to processes are parameterized, for example, $pos_p$ and $go_p$ are an observation and a transition for a process $p$ respectively. In OTS/CafeOBJ specifications, multitask systems are described with the import of loose module PID. For example, $pos_p$ is given by $pos : Pid \to \text{Sys} \to \text{Rat}$.

We give a system module $MS$ which imports data modules RAT, LABEL and PID for a signal control system with plural cars. Observations and initial state of module $MS$ are specified as follows:

\[
\begin{align*}
\text{eq } now\text{(init)} &= \emptyset . \\
\text{eq } going(P, \text{init}) &= \text{false}. \\
\text{eq } pos(P, \text{init}) &= \emptyset . \\
\text{eq } cs(P, \text{init}) &= \text{false}. \\
\text{eq } color\text{(init)} &= \text{gr}. \\
\text{eq } l\text{(init)} &= \emptyset .
\end{align*}
\]

where observations $pos$, $going$, $cs$ related to cars are parameterized. Equation $pos(P, \text{init}) = \emptyset$ means that the initial positions are zero for all cars $P$, for example. The definition of discrete transitions are modified as follows:

\[
\begin{align*}
\text{eq } c\text{-in}(P, S) &= (c_0 = \text{pos}(P, S) \text{ and color}(P, S) = \text{gr}) . \\
\text{eq } cs\text{(in}(P, S)) &= \text{false if } c\text{-in}(P, S). \\
\text{eq } cs(P', \text{in}(P, S)) &= P' = P \text{ or } cs(P', S) .
\end{align*}
\]

In the third equation above, term $cs(P', \text{in}(P, S))$ means that the value of $cs_{p'}$ at the result state of applying $in_p$ to state $s$ for processes $p$ and $p'$. Thus, when $p = p'$, it is true otherwise it is unchanged.

The effective condition of $tick$, should check all invariants for all processes. First, we parameterize $c\text{-tick}(P, X, S)$ by processes $P$ as follows:

\[
\begin{align*}
\text{eq } c\text{-tick}(P \to (P, X, S) &= (cs(P, S) \text{ and going}(P, S) \implies \\
&\text{cs_0 <= pos}(P, S) + X \text{ and } pos(P, S) + X <= cs_1 \text{ and } \\
&\text{and (cs_1 < pos}(P, S) + X \text{ implies not cs_0}(P, S)).}
\end{align*}
\]

We give a way to describe multitask hybrid systems in OTS/CafeOBJ specifications which denote hybrid automata without fixed numbers of processes. We introduce a specification $P\text{SET}$ of a set of processes:

\[
\begin{align*}
\text{mod* } P\text{SET} &= \{ \text{Pid} \in P\text{SET} \} \\
\text{op } _\_ &: P\text{SET} \to P\text{SET} (\text{assoc comm idem}) \\
\text{op } \text{nil} &\to P\text{SET} \text{ pred } _\text{in}: \text{Pid} \to P\text{SET} \\
\text{vars } P \in \text{Pid } \text{ var } P : P\text{SET} \\
\text{eq } (P \in (P \text{PS})) &= \text{true}. \\
\text{eq } (P \in \text{nil}) &= \text{false}. \\
\text{eq } (P \in Q) &= (P = Q). \\
\text{eq } (P \in (Q \text{PS})) &= (P = Q) \text{ or } (P \in \text{PS}).
\end{align*}
\]

Sort $P\text{SET}$ is declared as a super sort of $P\text{id}$, and the sequence of two elements of $P\text{SET}$ is also an element of $P\text{SET}$. Thus, a sequence of $P\text{id}$ is a term of $P\text{SET}$, for example, $p1 \ p2 \ p3$ is a term of $P\text{SET}$ when $p1$, $p2$ and $p3$ are terms of $P\text{id}$. Operation $\text{in}_\_\_$ denotes the membership predicate on $P\text{SET}$.

We also introduce an observation $ps$ which is a set of active processes. A process becomes active when it moves. After a process becomes active, it is active until it stops. The initial value of $ps$ is empty. When a car $p$ starts to move, $ps$ is updated by $p p\text{s}$. The following is a part of description related to $ps$:

\[
\begin{align*}
\text{eq } ps\text{(init)} &= \text{nil}. \\
\text{eq } ps(go(P, S)) &= P \text{ ps}(S).
\end{align*}
\]

The effective condition $c\text{-tick}$ is defined for $ps$. The effective condition of $tick$, is defined on $ps$ inductively:

\[
\begin{align*}
\text{eq } c\text{-tick}(\text{nil}(X, S)) &= \text{true}. \\
\text{eq } c\text{-tick}(P \ P\text{SET}, X, S) &= c\text{-tick}(P \to (P, X, S)) \text{ and } \\
&\text{c-tick}(PS, X, S). \\
\text{eq } c\text{-tick}(X, S) &= 0 <= X \text{ and } X <= cs_1 - cs_0 \text{ and } \\
&\text{c-tick}(ps(S), X, S). 
\end{align*}
\]

For example, $c\text{-tick}(p, q, r, x, s) = c\text{-tick}(p, x, s)$ and $c\text{-tick}(q, x, s)$ and $c\text{-tick}(r, x, s)$. 

117
IV. Verification of Multitask Hybrid Systems

In this section we give a formal proof of the safety property. We declare the relationship between constants $c_{s0}, c_{s1}$ and $t_0$ such that $0 < c_{s0} < c_{s1}$ and $c_{s1} - c_{s0} \leq t_0$. Then, the specification denotes all models satisfying the above condition including the test model in the previous section. First, we give a state predicate $inv_1(p,s)$ such that the car $p$ does not exist in the critical section when the signal is red at $s$.

$$eq \begin{align*}
    inv_1(P,S) &= not (color(S) = re \ and \\
    c_0 < pos(P,S) \ and \ pos(P,S) < cs1).
\end{align*}$$

If we prove $inv_1(p,s)$ for all processes $p$ and all reachable state $s$ from the initial state, the safety property holds. We denote the set of all reachable states by $RS$. We prove $\forall s \in RS. \forall p \in P_{id}. \, inv_1(p,s)$ by the induction on structure of reachable terms.

A Proof passage for the induction on the reachable state space

As the induction basis, we apply the reduction command to $inv_1(p,init)$ to prove the initial state to satisfy $inv_1$, where $p$ is a fresh constant as an arbitrary element. CafeOBJ interpreter returns true, which implies the induction basis holds, that is, $inv_1$ holds at the initial state.

In the induction step, we prove that each transition preserves the state predicate. We first assume an arbitrary state $s$ satisfies $inv_1(p,s)$ for all processes $p$. Then, under the assumption we prove $inv_1(p,s')$ for the state $s'$ obtained by any transition. The following is a template module for the induction step.

$$eq \begin{align*}
    istep1(P:Pid) &= inv1(P,s) \implies inv1(P,s').
\end{align*}$$

The proposition $istep1(P)$ means that the proposition $inv_1(P,s')$ holds under the induction hypothesis $inv_1(P,s)$. The following is a part of the induction steps with respect to $tick_i$, where $s'$ is obtained by applying $tick_i$ to $s$.

$$eq \begin{align*}
    s' &= tick(t1,s). \quad red \ \ istep1(p).
\end{align*}$$

If the above reduction returns true, it guarantees that implication $inv_1(p,s) \Rightarrow inv_1(tick_i(p,s))$ holds. Unfortunately, it returns neither true nor false for the above proof passage. We need to give more information for proofs. A typical proof strategy is a case splitting by the effective condition as follows:

$$eq \begin{align*}
    c-tick(t1,s) &= false. \quad eq \ s' &= tick(t1,s). \quad red \ \ istep1(p).
\end{align*}$$

If the above two proof passages both return true, the original proof passage is satisfied, since $(c - tick(t1,s) \Rightarrow istep1(p)) \land (\neg c - tick(t1,s) \Rightarrow istep1(p)) \Rightarrow istep1(p)$. If it does not so, we proceed case-splitting more.

B Lemma introduction

Repeating the process of case splitting, we may face false as a returned value of the reduction command. The following is such an example.

$$eq \begin{align*}
    cs(p,s) &= true. \quad ... \quad eq \ cs0 &= pos(p,s) = false. \quad eq \ s' &= tick(t1,s). \quad red \ istep1(p).
\end{align*}$$

Since $c_{s_p}$ is true, the position $pos_{s_p}$ is greater than or equal to $c_{s_0}$. The equation $cs0 = pos(p,s) = false$ contradicts to it. Such a proof passage represents unreachable states.

For such a case, we introduce another appropriate safety property, called a lemma. We add the following lemma to the module INV and ISTEP:

$$eq \begin{align*}
    inv3(P,S) &= cs(P,S) \implies \neg pos_{p} \implies pos_{p} = false. \quad eq \ istep3(P) &= inv3(P,s) \implies inv3(P,s').
\end{align*}$$

The lemma $inv3(P,S)$ denotes that the position of car $p$ is between $c_{s_0}$ and $c_{s_1}$ whenever $c_{s_p}$ is true. By replacing the reduction command by adding the lemma, we obtain true for the above proof passage.

$$eq \begin{align*}
    red \ \ inv3(p,s) \implies istep1(p).
\end{align*}$$

Proceeding the case splitting and introducing lemma, we obtain true for all the remaining proof passages for $inv_1$.

C Proving lemma

We proved invariant $inv_1$ for all states reachable from the initial state by the induction scheme under the assumption of some lemmata. To complete the proof, we need to show (1) the lemmata hold for the initial state, and (2) the lemmata hold for result state of applying every transition to states satisfying $inv_1$ and the lemmata. In the other words, we make a proof score of the conjunction of $inv_1 \land \cdots \land inv_n$, where the induction base is represented by $inv_1(p,init) \land \cdots \land inv_n(p,init)$ and the induction step is represented by $inv_1(p,s') \land \cdots \land inv_n(p,s')$ under the induction hypothesis $inv_1(p,s) \land \cdots \land inv_n(p,s)$. Note that in the previous section, we prove $inv_3(p,s) \Rightarrow (inv_1(p,s) \Rightarrow inv_3(p,s'))$. The formula is equivalent to $(inv_1(p,s) \land inv_3(p,s)) \Rightarrow inv_3(p,s')$. If we prove $(inv_1(p,s) \land inv_3(p,s)) \Rightarrow inv_3(p,s')$ for lemma $inv_3$, we obtain $(inv_1(p,s) \land inv_3(p,s)) \Rightarrow inv_1(p,s') \land inv_3(p,s')$. 

118
To complete a proof of inv1, we make seven lemmas and 136 proof passages\(^1\), all of which return true. The following is the declaration of the lemmata inv2, inv3, inv4, inv5, inv6 and inv7.

\[
\begin{align*}
eq \text{inv1}(P,S) & = \neg (\text{color}(S) = \text{re} \land \text{cs0} < \text{pos}(P,S) \land \text{pos}(P,S) < \text{cs1}) . \\
eq \text{inv2}(P,S) & = \neg (\text{cs}(P,S) \land \text{pos}(P,S) < \text{cs1} \land \text{color}(S) = \text{re}) . \\
eq \text{inv3}(P,S) & = \text{cs}(P,S) \implies \text{cs0} < \text{pos}(P,S) \land \text{pos}(P,S) < \text{cs1} . \\
eq \text{inv4}(P,S) & = \text{cs}(P,S) \land \text{pos}(P,S) < \text{cs1} . \\
eq \text{inv5}(P,S) & = \text{cs}(P,S) \land \text{pos}(P,S) < \text{cs1} \land \text{color}(S) = \text{gr} \implies \text{cs1} - \text{pos}(P,S) \leq \text{l}(S) - \text{now}(S) . \\
eq \text{inv6}(P,S) & = \text{cs}(P,S) \land \text{cs0} = \text{pos}(P,S) \land \text{going}(P,S) \implies \text{P in ps}(S) . \\
eq \text{inv7}(P,S) & = \text{cs0} < \text{pos}(P,S) \land \text{pos}(P,S) < \text{cs1} \implies \text{cs}(P,S) .
\end{align*}
\]

V. Related work

There are several tools for analyzing and/or verifying hybrid systems: MATLAB & Simulink\(^2\) HSolver\(^3\), HyTech\(^4\), KeYmaera\(^5\), PHAVer\(^6\), and so on. See the literature [5] for more details. One of the most relevant tools to our study is Maude a language and tool supporting specification description and verification based on rewriting logic [6]. Both Maude and CafeOBJ are algebraic specification languages and support user-defined abstract data type specifications, which is an advantage over the other tools for hybrid systems. Real-time Maude [7] is an extension of Maude which supports formal specification and analysis of real-time and hybrid systems. HI-Maude [8] is another extension of Maude which deals with a wider range of hybrid systems, called interacting hybrid systems. System modules in Maude are based on rewriting logic, where systems transitions are described by rewrite rules. Verification in Maude is based on exhaustive searching for reachable spaces obtained by the rewrite rules. In Maude, systems with discrete and continuous variables can be described, however, only discrete time domains obtained by time sampling strategies can be verified by search and model checking.

One of our advantages against these model checking approaches is that proof scores guarantee that verified properties hold for an arbitrary number of multiple processes. To make state spaces finite, model-checking approaches should restrict the size of the system to finite. Although model checking is fully-automated, the proof score method is semi-automated and needs a human interaction to complete proofs. In the literature [9], automated support of making proof scores has been proposed. A case splitting phase may be automated, however, lemma discovery is heuristic and not easy to be automated.

VI. Conclusion

We described an observational transition system of a simple signal control system with plural cars as an example of multitask hybrid systems, and verified some safety property by the proof score method. One of our future work is to apply the proposed method to practical applications of multitask hybrid systems, such as real-time operating systems, automotive control systems, intelligent transport systems, and so on.

References


---

\(^{1}\) Besides them, we need to add some lemmata which can be proved without induction on reachable states, e.g. eq lemma1(P,X,S) = (P in ps(S)) and c-tick(X,S) implies c-tick(P,X,S). We proved it by the induction on structure of terms of sort PSet.

\(^{2}\) https://www.mathworks.com/products/simulink.html

\(^{3}\) http://hsolver.sourceforge.net/

\(^{4}\) https://ptolemy.berkeley.edu/projects/embedded/research/hytech/

\(^{5}\) http://symboalnis.com/info/KeYmaera.html

\(^{6}\) http://www-verimag.imag.fr/%7Efrehse/phaver_web/index.html
Abstract—Proof scores are programs written in an algebraic specification language, such as CafeOBJ, to conduct formal verification. Thus, the proof score approach to formal verification (PSA2FV) can be regarded as a kind of proving by programming and then flexible. PSA2FV, however, is subject to human errors. To address the issue, a proof assistant called CiMPG was developed for CafeInMaude, the world’s second implementation of CafeOBJ. Furthermore, a proof generator called CiMPA was developed to benefit from the strong points of both PSA2FV and CiMPA. Although some case studies have been conducted with CiMPG, it is necessary to do some more. The present paper reports on case studies in which it is formally verified that two authentication protocols enjoy desired properties with CiMPG.

Keywords—algebraic specification language; proof assistant; proof generator; authentication protocol

I. INTRODUCTION

Theorem proving that systems enjoy some desired properties by writing proof scores have been intensively used. This approach uses observational transition systems (OTSs) as state machines to formalize systems. Then, the OTSs are specified in CafeOBJ, which is a formal specification language. Formal verification is conducted by writing what is called “proof scores” in CafeOBJ and executing them with CafeOBJ. Although writing proof scores is flexible to conduct formal verification, the proof may contain some flaws since proof scores are subject to human errors (e.g., users may overlook some cases during the proof).

CafeInMaude is the second implementation in Maude of CafeOBJ in addition to the original implementation in Common Lisp, where Maude is a sibling language of CafeOBJ. CafeInMaude introduces CafeOBJ specifications into the Maude system. It comes with two extension tools CafeInMaude Proof Assistant (CiMPA) and CafeInMaude Proof Generator (CiMPG). CiMPA is a proof assistant that allows users to write proof scripts in order to prove invariant properties on their CafeOBJ specifications. CiMPG provides a minimal set of annotations for identifying proof scores and generating CiMPA scripts for these proof scores. Using CiMPA to develop the formal verification by writing proof scripts can help us to avoid the flaw made by human users as in the proof score approach. However, it is often the case that CiMPA is not flexible enough to conduct formal verification. CiMPG allows users to combine the flexibility of the proof score approach and the reliability of CiMPA. Given proof scores that should be slightly annotated, CiMPG generates proof scripts for CiMPA. Feeding the generated proof scripts into CiMPA, if CiMPA successfully discharges all goals, the proof scores are correct for the goals.

This paper presents the formal verification with CiMPA and CiMPG of two authentication protocols: Identity-Friend-or-Foe authentication protocol (IFF) and Needham-Schroeder-Lowe Public Key authentication protocol (NSLPK). The former is a simple protocol used to check if a principal (or an agent) is a member of a group. The latter is an advanced authentication protocol, which is a modification of NSPK protocol made by Lowe. We use CiMPA and CiMPG to formally verify that: (1) IFF enjoys the identifiable property, and (2) NSLPK enjoys the nonce secrecy property and one-to-many correspondence property.

Although it has been formally verified that NSLPK enjoys the nonce secrecy property with CiMPG, we are the first to formally verify that NSLPK enjoys the one-to-many correspondence property with CiMPA as well as CiMPA. IFF is a tiny protocol but nobody has formally verified that it enjoys a desired property with either CiMPA or CiMPG. All specifications and proofs presented in this paper are available at https://github.com/twmon14/fvap.

II. FORMAL VERIFICATION OF IFF

IFF is used to check if a principal is a member of a group. The IFF protocol can be described as the following two message exchanges:

Check \( p \rightarrow q : r \)

Reply \( q \rightarrow p : \varepsilon_k(r,q) \)

Each principal (or agent) such as \( p \) and \( q \) belongs to only one group. A symmetric key is given to each group, whose members share the key, and keys are different from group to group. If a principal \( p \) wants to check if a principal \( q \) is a member of the \( p \)'s group, \( p \) generates a fresh random number \( r \) and sends it to \( q \) as a Check message. On receipt of the message, \( q \) sends back to \( p \) a Reply message that consists of \( r \) and ID \( q \) encrypted by the symmetric key \( k \) of the \( q \)'s group. When \( p \) receives the Reply message, \( p \) tries to decrypt the ciphertext received with the symmetric key of the \( p \)'s group. If the decryption succeeds and the plaintext...
consists of \( r \) and \( q \), \( p \) then concludes that \( q \) is a member of the \( p \)'s group.

We suppose that the cryptosystem used is perfect, there is only one legitimate group, all members of the group are trustable, and there are also untrustable principals who are not members. Trustable principals exactly follow the protocol, but untrustable ones may do something against the protocol as well. The combination and cooperation of untrustable principals are modeled as the most general enemy (or intruder). The enemy gleans as much information as possible from messages flowing in the network and creates fake messages based on the gleaned information, provided that the enemy cannot break the perfect cryptosystem.

**A. Formal Specification of the Protocol**

We first declare the operator \( \text{enc} \) to specify the ciphertexts used in the protocol as follows:

\[
\begin{align*}
\text{op} & \quad \text{enc} : \text{Key} \times \text{Rand} \times \text{Prin} \rightarrow \text{Cipher} . \\
\text{op} & \quad k : \text{Cipher} \rightarrow \text{Key} . \quad \text{op} \quad r : \text{Cipher} \rightarrow \text{Rand} . \\
\text{op} & \quad p : \text{Cipher} \rightarrow \text{Prin} . \\
\end{align*}
\]

where \( \text{Key} \) is the sort (or type) representing symmetric keys, \( \text{Rand} \) is the sort denoting random numbers, \( \text{Prin} \) is the sort representing principals, and \( \text{Cipher} \) is the sort denoting ciphertexts. Given a key \( k \), a random number \( r \) and a principal \( p \), \( \text{enc}(k, r, p) \) denotes the ciphertext obtained by encrypting \( r \) and \( p \) with \( k \). Operators \( k \), \( r \) and \( p \) return the first, second and third arguments of \( \text{enc}(k, r, p) \), respectively.

We specify two messages \( \text{Check} \) and \( \text{Reply} \) by two operators \( \text{cm} \) and \( \text{rm} \) as follows:

\[
\begin{align*}
\text{op} & \quad \text{cm} : \text{Prin} \times \text{Prin} \times \text{Prin} \times \text{Rand} \rightarrow \text{Msg} . \\
\text{op} & \quad \text{rm} : \text{Prin} \times \text{Prin} \times \text{Prin} \times \text{Cipher} \rightarrow \text{Msg} . \\
\end{align*}
\]

where \( \text{Msg} \) is the sort denoting messages. The first, second and third arguments of each of \( \text{cm} \) and \( \text{rm} \) are the actual creator, the seeming sender and the receiver of the corresponding message. The first argument is meta-information that is only available to the outside observer and the principal that has sent the corresponding message, and that can not be forged by the enemy; while the remaining arguments may be forged by the enemy.

The network is modeled as a multiset of messages, in which the enemy can use as his/her storage. Any message that has been sent or put once into the network is supposed to be never deleted from the network because the enemy can replay the message repeatedly, although the enemy can not forge the first argument. Consequently, the empty network (i.e., the empty multiset) means that no messages have been sent.

The enemy tries to glean two kinds of values from the network, which are random numbers and ciphertexts. The collections of these values gleaned by the enemy are denoted by operators \( \text{rands} \) and \( \text{ciphers} \), which are declared as follows:

\[
\begin{align*}
\text{op} & \quad \text{rands} : \text{Network} \rightarrow \text{ColRands} . \\
\text{op} & \quad \text{ciphers} : \text{Network} \rightarrow \text{ColCiphers} .
\end{align*}
\]

where \( \text{Network} \) is the sort denoting networks, \( \text{ColRands} \) is the sort denoting collections of random numbers, and \( \text{ColCiphers} \) is the sort denoting collections of ciphertexts. \( \text{ciphers} \) is defined by the following equations:

\[
\begin{align*}
\text{eq} & \quad \text{ciphers}(\text{void}) = \text{false} . \\
\text{ceq} & \quad \text{ciphers}(M, \text{NW}) = \text{true} \quad \text{if} \quad \text{rm}(M) \quad \text{and} \quad C = \text{c}(M) . \\
\text{ceq} & \quad \text{ciphers}(M, \text{NW}) = \text{ciphers}(\text{NW}) \quad \text{if} \quad \text{not}(\text{rm}(M) \quad \text{and} \quad C = \text{c}(M)) .
\end{align*}
\]

where \( \text{void} \) denotes the empty multiset (or empty network), \( \text{rm} \) checks if a given message is a Reply message, \( \text{op} \) takes a Reply message as a parameter and returns its ciphertext (i.e., the fourth argument of \( \text{rm} \) operator), \( \text{nw} \) is an infix operator checking the existence of an element in a collection, and \( \text{ur} \), in \( M, \text{NW} \) denotes the data constructor of nonempty multisets. The equations say that a ciphertext \( C \) is available to the enemy iff there exists a Reply message whose content is \( C \). \( \text{rands} \) can be defined likewise.

Now, we are ready to specify the protocol. We use two observational functions \( \text{nw} \) and \( \text{ur} \) to observe the network and the set of used random numbers, respectively as follows:

\[
\begin{align*}
\text{op} & \quad \text{nw} : \text{Sys} \rightarrow \text{Network} . \quad \text{op} \quad \text{ur} : \text{Sys} \rightarrow \text{URands}.
\end{align*}
\]

where \( \text{Sys} \) is the sort denoting the state space of IFF, \( \text{URands} \) is the sort denoting the sets of random numbers.

We use five transitions together with one constant of \( \text{Sys} \) to represent an arbitrary initial state as follows:

\[
\begin{align*}
\text{op} & \quad \text{init} : \rightarrow \text{Sys} \quad \text{constr} . \\
\text{op} & \quad \text{sdcm} : \text{Sys} \times \text{Prin} \times \text{Prin} \times \text{Rand} \rightarrow \text{Sys} \quad \text{constr} . \\
\text{op} & \quad \text{sdm} : \text{Sys} \times \text{Prin} \times \text{Rand} \rightarrow \text{Sys} \quad \text{constr} . \\
\text{op} & \quad \text{fkcm1} : \text{Sys} \times \text{Prin} \times \text{Prin} \times \text{Rand} \rightarrow \text{Sys} \quad \text{constr} . \\
\text{op} & \quad \text{fkcm2} : \text{Sys} \times \text{Prin} \times \text{Rand} \rightarrow \text{Sys} \quad \text{constr} .
\end{align*}
\]

\( \text{sdcm} \) and \( \text{sdm} \) formalize sending \( \text{Check} \) and \( \text{Reply} \) messages exactly following the protocol, respectively. The remaining actions \( \text{fkcm1}, \text{fkcm2}, \) and \( \text{fkrm2} \) are the enemy’s faking messages, which can be understood as follows:

- \( \text{fkcm1} \): a random number \( R \) is available to the enemy, the enemy fakes and sends a Check message using \( R \).
- \( \text{fkrm1} \): a ciphertext \( C \) is available to the enemy, the enemy fakes and sends a Reply message using \( C \).
- \( \text{fkrm2} \): a random number \( R \) is available to the enemy, the enemy fakes and sends a Reply message using \( R \).

\( \text{sdcm} \) is defined as follows:

\[
\begin{align*}
\text{ceq} & \quad \text{nw}(\text{sdcm}(S, P1, P2, R)) = (\text{cm}(P1, P1, P2, R), \text{nw}(S)) \quad \text{if} \quad c-\text{sdcm}(S, P1, P2, R) . \\
\text{ceq} & \quad \text{ur}(\text{sdcm}(S, P1, P2, R)) = (R \text{ ur}(S)) \quad \text{if} \quad c-\text{sdcm}(S, P1, P2, R) . \\
\text{ceq} & \quad \text{sdcm}(S, P1, P2, R) = S \quad \text{if} \quad \text{not} \quad c-\text{sdcm}(S, P1, P2, R) .
\end{align*}
\]

121
where \(c^{-}\text{sdcm}(S,P_1,P_2,R)\) is not \((R \notin \text{ur}(S))\).

The equations say that if \(c^{-}\text{sdcm}(S,P_1,P_2,R)\) is true (i.e., \(R\) has not been used), then the Check message \(\text{cm}(P_1,P_2,R)\) is put into the network \(\text{nw}(S)\). \(R\) is put into \(\text{ur}(S)\) in the state denoted by \(\text{sdcm}(S,P_1,P_2,R)\); if \(c^{-}\text{sdcm}(S,P_1,P_2,R)\) is false, nothing changes. The remaining transitions can be defined likewise.

### B. Formal Verification with CiMPA

One property of IFF we would like to confirm is whenever \(p\) receives a valid Reply message from \(q, q\) is always a member of the \(p\)'s group. Such property is called identifiable property in this paper. The property is specified as follows:

\[
op \text{inv1} : \text{Sys Prin Prin Prin Key Rand} \rightarrow \text{Bool}.

\[
eq \text{inv1}(S,P_1,P_2,P,R) = ((\neg(K = k(\text{enemy})))
\text{and rm}(P_1,P_2,P,\text{enc}(K,R,P_2))) \in \text{nw}(S))\] implies \((P_2 = \text{enemy}) = \text{true} \).

We describe how to prove that IFF enjoys the property by writing proof scripts and running with CiMPA. In the proof script, we need to use a lemma \(\text{inv2}\) that is as follows:

\[
op \text{inv2} : \text{Sys Key Rand} \rightarrow \text{Bool}.

\[
eq \text{inv2}(S,K,R) = (\text{enc}(K,R,\text{enemy}) \in \text{nw}(S))\] implies \((K = k(\text{enemy})) = \text{true} \).

where \(k(\text{enemy})\) denotes the symmetric key of the group to which the enemy belongs to.

The proof starts with the goals we need to prove:

\[	ext{open IFF}.

\[
\text{eq [iff1 :nonexec]} : \text{inv1}(S:\text{Sys},P:\text{Prin},
P_1:\text{Prin},P_0:\text{Prin},K:\text{Key},R:\text{Rand}) = \text{true} .

\[
\text{eq [iff :nonexec]} : \text{inv2}(S:\text{Sys},K:\text{Key},R:\text{Rand}) = \text{true} .
\]

where IFF is the module in which the specification of IFF together with \(\text{inv1}\) and \(\text{inv2}\) are available. :nonexec instructs CafeInMaude not to use the equations as rewrite rules.

Then, we select \(S\) with the command :ind on as the variable on which we start proving the goals by simultaneous induction:

\[
\text{ind on } (S:\text{Sys}) \text{ :apply(si)}
\]

The command :apply(si) starts the proof by simultaneous induction on \(S\), generating six goals for \(\text{fkcm1}, \text{fkrm2}, \text{init}, \text{sdcm},\) and \(\text{sdrm}\), where \(\text{si}\) stands for simultaneous induction. Each goal consists of two equations to prove, corresponding to \(\text{inv1}\) and \(\text{inv2}\). With the first goal for \(\text{fkcm1}\), we first apply theorem of constants by using the command :apply(tc). The command generates two sub-goals as follows:

1.1. \(\text{TC eq [iff1 :nonexec]} : \text{inv1}(\text{fkcm1}(S:\text{Sys},P_1\text{Prin},P_0\text{Prin},R\text{Rand}),P_1\text{Prin},P_0\text{Prin},K\text{Key},R\text{Rand}) = \text{true} .

1.2. \(\text{TC eq [iff :nonexec]} : \text{inv2}(\text{fkcm1}(S:\text{Sys},P_1\text{Prin},P_0\text{Prin},R\text{Rand}),K\text{Key},R\text{Rand}) = \text{true} .

The command :apply(tc) replaces CafeOBJ variables with fresh constants in goals. \(S\text{#Sys}, P\text{#Prin}, P_0\text{#Prin},\) and \(R\text{#Rand}\) are fresh constants introduced by :apply(si), while \(P\text{#Prin}, P_0\text{#Prin}, K\text{#Key},\) and \(R\text{#Rand}\) are fresh constants introduced by :apply(tc). To discharge goal 1-1, the following commands are first introduced:

\[
\text{def c1 = :ctf } (R\text{#Rand} \in \text{rands}(nw(S\text{#Sys}))).

\]

\[
\text{apply(c1)}
\]

Goal 1-1 is split into two sub-goals 1-1-1 and 1-1-2 correspond to when \(R\text{#Rand} \in \text{rands}(nw(S\text{#Sys}))\) holds and does not hold, respectively. Then, two sub-goals are discharged by the following commands:

\[
\text{def c2 = :ctf } (R\text{#Rand} \in \text{rands}(nw(S\text{#Sys}))).

\]

\[
\text{apply(c2)}
\]

\[
\text{imp [iff]} by \{K\text{Key} \leftarrow K\text{#Key};
P_0\text{Prin} \leftarrow P_0\text{#Prin}; P_1\text{Prin} \leftarrow P_1\text{#Prin} ;
P:\text{Prin} \leftarrow P\text{#Prin}; R:\text{Rand} \leftarrow R\text{#Rand} \}.

\]

\[
\text{apply(rd)}
\]

\[
\text{imp [iff]} by \{K\text{Key} \leftarrow K\text{#Key} ;
P_0\text{Prin} \leftarrow P_0\text{#Prin}; P_1\text{Prin} \leftarrow P_1\text{#Prin} ;
P:\text{Prin} \leftarrow P\text{#Prin}; R:\text{Rand} \leftarrow R\text{#Rand} \}.

\]

\[
\text{apply(rd)}
\]

The induction hypothesis is instantiated by replacing the variables with the fresh constants and the instance is used as the premise of the implication. For example, \(P_1\text{Prin}\) is replaced with \(P\text{#Prin}\). Then, :apply(rd) is used to check if the current goal can be discharged. Two goals 1-1-1 and 1-1-2 are discharged in this case. The current goal is changed to 1-2.

Goal 1-2 is split into two sub-goals and they are discharged by the following commands:

\[
\text{def c3 = :ctf } (C\text{#Cipher} \in \text{ciphers}(nw(S\text{#Sys}))).

\]

\[
\text{apply(c3)}
\]

\[
\text{def c4 = :ctf } (\text{eq } P\text{#Prin} = \text{enemy}).

\]

\[
\text{def c5 = :ctf } (\text{eq } P\text{#Prin} = P\text{#Prin}).

\]

\[
\text{def c6 = :ctf } (\text{eq } P_0\text{Prin} = P_0\text{Prin}).

\]

\[
\text{def c7 = :ctf } (\text{eq } K\text{#Cipher} = K\text{#Key} ).

\]

\[
\text{def c8 = :ctf } (\text{eq } R\text{#Rand} = R\text{#Rand} ).

\]

\[
\text{def c9 = :ctf } (\text{eq } P\text{#Cipher} = P\text{#Prin} ).

\]

\[
\text{def c10 = :ctf } (\text{eq } K\text{#Key} = k(\text{enemy}) ).

\]

\[
\text{apply(c3)} \text{ :apply(c4)} \text{ :apply(c5)} \text{ :apply(c6)} \text{ :apply(c7)} \text{ :apply(c8)} \text{ :apply(c9)} \text{ :apply(c10)}
\]

We have all done with goal 1, the current goal moves to 2. With goal 2, we first introduce the following commands to conduct case splitting:

\[
\text{def c3 = :ctf } (C\text{#Cipher} \in \text{ciphers}(nw(S\text{#Sys}))).

\]

\[
\text{def c4 = :ctf } (\text{eq } P\text{#Prin} = \text{enemy}).

\]

\[
\text{def c5 = :ctf } (\text{eq } P\text{#Prin} = P\text{#Prin}).

\]

\[
\text{def c6 = :ctf } (\text{eq } P_0\text{Prin} = P_0\text{Prin}).

\]

\[
\text{def c7 = :ctf } (\text{eq } K\text{#Cipher} = K\text{#Key}).

\]

\[
\text{def c8 = :ctf } (\text{eq } R\text{#Rand} = R\text{#Rand}).

\]

\[
\text{def c9 = :ctf } (\text{eq } P\text{#Cipher} = P\text{#Prin}).

\]

\[
\text{def c10 = :ctf } (\text{eq } K\text{#Key} = k(\text{enemy})).

\]

\[
\text{apply(c3)} \text{ :apply(c4)} \text{ :apply(c5)} \text{ :apply(c6)} \text{ :apply(c7)} \text{ :apply(c8)} \text{ :apply(c9)} \text{ :apply(c10)}
\]

Case splittings are carried out based on one Boolean term and seven equations. The first sub-goal in which the Boolean term is true and seven equations hold can be discharged:
The lemma \textit{inv2} as a lemma:

\begin{verbatim}
:apply (c11) :apply (c12)
:proof (iff) \{K:Key <- K@Key ; R:Rand <- \&0Rand ;\}
:apply (rd)
\end{verbatim}

The lemma \textit{inv2} is instantiated by replacing the variables K:Key and R:Rand with the fresh constants K@Key and \&0Rand, and the instance is used as the premise of the implication. The induction hypothesis is instantiated by replacing the variables with the fresh constants, and the instance is used as the premise of the implication. Then, :apply (rd) is used to discharge the current goal. The remaining sub-goals of 2 can be discharged directly without using any lemma. The remaining goals from 3 to 6 can be discharged likewise.

\subsection*{C. Formal Verification with CiMPG}

The following is the proof score for the case corresponding to goal 1-1-1 in the last section:

\begin{verbatim}
open IFF .
:proof (iff) 
op s : -> Sys . ops a b c d e : -> Prin .
op k : -> Key . ops r1 r2 : -> Rand .
eq (r2 \in \nands (nw(s))) = true .
red invl(s,a,b,c,k,r1) implies invl(fkml(s,d,e,r2),a,b,c,k,r1) .
close
\end{verbatim}

where \textit{open} makes the module \textit{IFF} available, \textit{close} stops the use of the module and \textit{red} reduces (computes) the given term. \textit{s} and \textit{k} correspond to \textit{S\#Sys} and \textit{K\#Key} in the last section, respectively. a, b, c, d, and e correspond to \textit{P\#Prin}, \textit{P\#Prin}, \textit{P\#Prin}, \textit{P\#Prin}, and \textit{P\#Prin}, respectively. \textit{r1} and \textit{r2} correspond to \textit{R\#Rand} and \textit{R\#Rand}, respectively. The details of the proof score approach as well as how to write proof scores to conduct formal verification can be found in paper \cite{1}. In comparison with proof scripts, proof scores are often easier to understand for human users, and writing proof scores are also more flexible than writing proof scripts. That is the reason why when conducting formal verification, we prefer to write proof scores rather than proof scripts. However, because of the flexibility, proof scores are subject to human errors. For example, during the verification users may overlook some cases, leading to the flaw verification.

After writing proof scores that \textit{IFF} protocol enjoys the property, we can confirm that the proof scores are correct by doing the formal verification with CiMPA as described in the last section. Although we are able to conduct the formal verification with CiMPA once we have completed formal verification by writing proof scores in CafeOBJ, it would be preferable to automatically confirm the correctness of proof scores. CiMPG makes it possible to automatically confirm the correctness of proof scores by generating proof scripts for CiMPA from the proof scores.

To use CiMPG, we need to add :id(iff) into each open-close proof score fragment. For example, the open-close fragment shown above becomes as follows:

\begin{verbatim}
open IFF .
:proof (iff) 
open IFF .
:proof (iff) 
close
\end{verbatim}

where \textit{iff} is just an identifier, can be replaced by another one that is more preferred.

Feeding the annotated proof scores into CiMPG, CiMPG generates the proof script for CiMPA. The generated proof script is quite similar to the one written manually. Feeding the generated proof script into CiMPA, CiMPA discharges all goals, confirming that the proof scores are correct.

\section*{III. Formal verification of NSLPK}

\textbf{NSLPK} \cite{6} is a modification of NSPK authentication protocol \cite{7} made by Lowe. The NSLPK protocol can be described as the following three message exchanges:

\begin{verbatim}
Init \quad p \rightarrow q : \varepsilon_q(n_p, p) 
Resp \quad q \rightarrow p : \varepsilon_p(n_q, p) 
Ack \quad p \rightarrow q : \varepsilon_q(n_p) 
\end{verbatim}

Each principal such as \textit{p} and \textit{q} has a pair of keys: public and private keys. \(\varepsilon_p(m)\) denotes the ciphertext obtained by encrypting the message \(m\) with the principal \textit{p}'s public key. \(n_p\) is a nonce (a random number) generated by principal \textit{p}. A nonce is a unique and non-guessable number that is used only one time. Again, we also suppose that the cryptosystem used is perfect.

\subsection*{A. Formal Specification of the Protocol}

We introduce the following three operators to represent the ciphertexts used in the protocol:
where \( \text{Nonce} \) is the sort denoting the nonce numbers; Cipher1, Cipher2, and Cipher3 are the sorts denoting three kinds of ciphertexts contained in Init, Resp, and Ack messages, respectively. Given principals \( p, q \) and a nonce \( n_p \) term \( \text{enc1}(q, n_p, p) \) denotes the ciphertext \( \text{cipher1}(n_p, p) \) obtained by encrypting \( n_p \) and \( p \) with principal \( q \)'s public key. \( \text{enc2} \) and \( \text{enc3} \) can be understood likewise.

We specify three messages used in NSLPK as follows:

\[
\begin{align*}
\text{op m1 : Prin Prin Prin Cipher1} & \rightarrow \text{Msg} \\
\text{op m2 : Prin Prin Prin Cipher2} & \rightarrow \text{Msg} \\
\text{op m3 : Prin Prin Prin Cipher3} & \rightarrow \text{Msg}
\end{align*}
\]

\( \text{Msg} \) as well as the first three arguments of each operator can be understood as in the specification of IFF explained in the last section.

The intruder tries to glean four kinds of values from the network, which are nonces and three kinds of ciphertexts. Then, we use following four operators to denote those values:

\[
\begin{align*}
\text{op cnonce : Network} & \rightarrow \text{ColNonce} \\
\text{op cenc1 : Network} & \rightarrow \text{ColCipher1} \\
\text{op cenc2 : Network} & \rightarrow \text{ColCipher2} \\
\text{op cenc3 : Network} & \rightarrow \text{ColCipher3}
\end{align*}
\]

where \( \text{Network} \) is the sort denoting networks (i.e., multisets of messages) and \( \text{ColX} \) is a sort denoting collections of values corresponding to the sort \( X \). The equations defining \( \text{cenc1} \) are as follows:

\[
\begin{align*}
\text{eq E1 \& in cenc1(void)} & = \text{false} \\
\text{ceq E1 \& in cenc1(M,NW)} & = \text{true if m1?(M)} \text{ and not(key(cipher1(M))) = intruder)} \text{ and E1 = cipher1(M)} \\
\text{ceq E1 \& in cenc1(NW)} & = \text{E1 \& in cenc1(NW) if not(m1?(M)) and E1 = cipher1(M)} \text{ and not(key(cipher1(M)) = intruder)}
\end{align*}
\]

where \( E1 \) is a CafeOBJ variable of Cipher1. \( m1? \) checks if a given message is an Init message. Operator \( \text{cipher1} \) takes an Init message as an argument and returns its ciphertext (i.e., the fourth argument of \( m1 \) operator). Operator \( \text{cipher} \) takes a ciphertext as an argument and returns the principal in which the ciphertext is encrypted with its public key. void, M, NW, as well as \( (M, NW) \) can be understood as explained in the last section. The equations say that a ciphertext \( E1 \) is available to the intruder if there exists an Init message whose content is \( E1 \) and \( E1 \) is not encrypted by the intruder's public key. Let us note that, if \( E1 \) is encrypted by the intruder's public key, \( E1 \) can be rebuilt by the intruder. \( \text{cnonce}, \text{cenc2}, \text{cenc3} \) can be defined likewise.

We use two observers, nine transitions, together with one constant that represents an arbitrary initial state to specify NSLPK as follows:

\[
\begin{align*}
\text{op ur : Sys} & \rightarrow \text{URand} \\
\text{op nw : Sys} & \rightarrow \text{Network} \\
\text{op init :} & \rightarrow \text{Sys (constr)}
\end{align*}
\]

\( \text{op sdml : Sys Prin Prin Rand} \rightarrow \text{Sys (constr)} \)

\( \text{op sdm2 : Sys Prin Rand Msg} \rightarrow \text{Sys (constr)} \)

\( \text{op sdm3 : Sys Prin Rand Msg Msg} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm11 : Sys Prin Prin Cipher1} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm12 : Sys Prin Prin Nonce} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm21 : Sys Prin Prin Cipher2} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm22 : Sys Prin Prin Nonce Nonce} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm31 : Sys Prin Prin Cipher3} \rightarrow \text{Sys (constr)} \)

\( \text{op fkm32 : Sys Prin Prin Nonce} \rightarrow \text{Sys (constr)} \)

where \( \text{URand} \) is the sort denoting sets of random numbers. \( \text{ur, nw, and init} \) can be understood as in the last section. The first three transitions formalize sending messages exactly following the protocol, while the remaining formalize the intruder's faking messages, which can be understood as follows:

- \( \text{fkm11}, \text{fkm21}, \text{and fkm31: a ciphertext C is available to the intruder, the intruder fakes and sends an Init, or Resp, or Ack message using C, respectively.} \)
- \( \text{fkm12 and fkm32: a nonce N is available to the intruder, the intruder fakes and sends an Init or Ack message using N, respectively.} \)
- \( \text{fkm22: two nonces N1 and N2 are available to the intruder, the intruder fakes and sends a Resp message using N1 and N2.} \)

Let \( S \) be a CafeOBJ variable of Sys, and \( P & Q \) are CafeOBJ variables of Prin. \( fkm11 \) is defined as follows:

\[
\begin{align*}
\text{eq ur(fkm11(S,P,Q,E1))} & = \text{ur(S)} \\
\text{ceq nw(fkm11(S,P,Q,E1))} & = \text{ml(intruder,P,Q,E1)} \\
\text{ceq fkm11(S,P,Q,E1)} & = \text{S if not c-fkm11(S,P,Q,E1)} \\
\text{where c-fkm11(S,P,Q,E1)} & = \text{E1 \& in cenc1(nw(S)), intruder is a constant of Prin denoting the intruder. The equations say that if c-fkm11(S,P,Q,E1) is true, then the Init message ml(intruder,P,Q,E1) is put into the network nw(S), ur(S) does not change in the state denoted by fkm11(S,P,Q,E1); if c-fkm11(S,P,Q,E1) is false, nothing changes. The remaining transitions can be defined likewise.} \)
\]

**B. Formal Verification with CiMPA and CiMPG**

There are two properties of NSLPK that we would like to verify namely nonce secrecy property and one-to-many correspondence property. The former says that all nonces available to the intruder are those created by the intruder or those created for the intruder. Let \( N \) be a CafeOBJ variable of Nonce, we specify the nonce secrecy property as follows:

\[
\text{eq invi30}(S,N) = (N \& in cnonce(nw(S))) \implies (creator(N) = intruder or}
\]
where $E2$ and $Q1$ are CafeOBJ variables of Prin, $R$ is a CafeOBJ variable of Rand. $inv170$ says that whenever $P$ successfully sent an Init message to $Q$, and received a corresponding Resp seemingly from $Q$, the principal that $P$ is communicating with is really $Q$ even though there are malicious principals (e.g., $Q1$). $inv180$ can be understood likewise.

To verify the nonce secrecy property, we prove that $inv130$ is an invariant of the OTS formalizing NSLPK. The formal verification is also conducted in two ways: by writing proof scripts with CiMPA and by using CiMPG to generate proof scripts from proof scores. Both of them require the use of the following lemmas:

$$eq\ inv170(S,P,Q,Q1,R,N) = (\neg(P = intruder) \text{ and } m1(P,P,Q,enc1(Q,n(P,Q,R),P)) \in nw(S) \text{ and } m2(Q1,Q,P,enc2(P,n(P,Q,R),N,Q)) \in nw(S)) .$$

$$eq\ inv180(S,P,Q,P1,R,N) = (\neg(Q = intruder) \text{ and } m2(Q,Q,P,enc2(P,n(P,Q,R),N,Q)) \in nw(S) \text{ and } m3(P1,P,Q,enc3(Q,n(Q,P,R))) \in nw(S)) .$$

where $P1$ and $Q1$ are CafeOBJ variables of Prin, $R$ is a CafeOBJ variable of Rand. $inv170$ says that whenever $P$ successfully sent an Init message to $Q$, and received a corresponding Resp seemingly from $Q$, the principal that $P$ is communicating with is really $Q$ even though there are malicious principals (e.g., $Q1$). $inv180$ can be understood likewise.

In each way of verification, what we need to do is quite similar to what we have described in the last section with formal verification of IFF. However, with CiMPG, we also need to make some modifications to the existing proof scores. Let us consider an example in which we want to split the current case into two sub-cases: (1) message $m$ is in $nw(s)$, which is the network of the current state, and (2) $m$ is not in $nw(s)$. CafeOBJ allows us to write proof scores to conduct case splitting by introducing two equations: (i) $nw(s) = (m, nw')$ to characterize (1) and (ii) $m \not\in nw(s) = false$ to characterize (2), where $nw'$ is a constant denoting an arbitrary network (or list of messages). With CiMPA, if we declare equation (i) and apply for case splitting, then it will automatically split the current goal into two sub-goals in which (i) holds in the first sub-goal, while it does not hold in the second one. Thus, the second sub-goal is characterized by the equation $(nw(s) = (m, nw')) = false$. In this sub-goal, it does not guarantee that $m$ is not in $nw(s)$ since $m$ can be in $nw'$. CiMPG also cannot recognize that the use of two equations (i) and (ii) for case splitting is correct. In the existing proof scores of formal verification of NSLPK, there are many times in which case splitting is “flexibly” applied in the same way as based on two equations (i) and (ii) mentioned above. This flexible case splitting is an advantage of the CafeOBJ/proof score method but also is a disadvantage because we need to ensure that the equations used for case splitting cover every case and do not overlap each other. However, to make it possible for CiMPG to generate the proof scripts, the existing proof score needs to be modified. With the example mentioned above, two equations used for case splitting should be $m \not\in nw(s) = true$ and $m \not\in nw(s) = false$.

IV. Conclusion

This paper has presented the formal verifications with proof assistant CiMPA and with proof generator CiMPG. In comparison with the proof score approach, each verification method has advantages as well as disadvantages. While proof scores are flexible to write, they are subject to human errors since human users can overlook some cases during the verification. The proof scripts are reliable, but they are not easy to develop, especially with non-expert users. CiMPG combines the flexibility of the proof score approach and the reliability of CiMPA. However, it often takes time for CiMPG to generate proof scripts when the size of input proof scores is large. Two case studies are presented in which we formally verify that IFF protocol enjoys the identifiable property, and NSLPK enjoys the nonce secrecy and one-to-many correspondence properties.

References

Abstract—The second and third authors of the present paper have formally verified that A-Anderson protocol, which is an abstract version of Anderson mutual exclusion protocol, enjoys the mutual exclusion property in their previous work. The reason why they did not conduct formal verification with the original version of Anderson but with A-Anderson instead is that Anderson uses a finite boolean array and the modulo (or remainder) operation of natural numbers, causing the challenge to conduct formal verification in a sense of theorem proving. Since then, we have successfully completed formal verification with Anderson to which an auxiliary variable is introduced. The protocol is specified in CafeOBJ, an algebraic specification language, and it is formally verified that the protocol enjoys the property with CafeOBJ. The auxiliary variable does not change the behavior of Anderson. We then conclude that Anderson enjoys the mutual exclusion property by proving that the property is an invariant of the specification. We also informally discuss why it is necessary to introduce auxiliary variables so that we can successfully complete formal verification with some protocols or systems.

Keywords—algebraic specification language; mutual exclusion protocol; auxiliary variable; proof score

I. INTRODUCTION

Mutual exclusion is the problem such that at most one thread, process, node, or any execution entity is allowed to enter its critical section to use some shared resources, such as shared memory in concurrent and/or distributed systems. Mechanisms or protocols that solve the problem are called mutual exclusion protocols. Anderson protocol (or Anderson) [1] is a mutual exclusion protocol. Thus, the most important property the protocol should satisfy is the mutual exclusion property. It is, however, challenging to formally verify that Anderson protocol enjoys the mutual exclusion property, in a sense of theorem proving. The reason is that its algorithm uses a finite array and the modulo operation of natural numbers. In the paper [2], the second and third authors of the present paper have introduced an abstract version of Anderson, which is called A-Anderson protocol (or A-Anderson), and formally verified that A-Anderson enjoys the mutual exclusion property.

Although as mentioned in the paper [2], the authors have successfully proved that Anderson enjoys the mutual exclusion property by showing that there exists a simulation relation from Anderson to A-Anderson, and such simulation preserves the property, directly proving that Anderson enjoys the property still interests us. In Anderson, each process is located at one of three locations: rs (Remainder Section), ws (Waiting Section), or cs (Critical Section). Initially, each process is located at rs, and when a process wants to enter cs, it first moves to ws from rs. By introducing an auxiliary variable to record a collection of processes currently located at ws or cs, we have successfully completed the proof that Anderson satisfies the mutual exclusion property without using the abstract version A-Anderson but with an auxiliary variable introduced. This modification only records information in the current and past states but does not affect the current or future values of any other variables in the algorithm. Therefore, we can guarantee that adding the auxiliary variable does not change the behavior of Anderson. Originally, we got stuck in the formal verification of Anderson because the proof requires a lemma that is obvious but so tough to prove. Introducing the auxiliary variable helps us to accomplish the proof of that lemma, leading to the complete formal verification. The proof of the lemma has not yet been completed as of the paper submission without introducing any auxiliary variables, though.

Our verification in this paper uses observational transition systems (OTSs) [3] as state machines. The OTS formalizing Anderson is specified in CafeOBJ [4], which is a formal specification language. Then, in the specification, we introduce an auxiliary variable to store a list of processes currently located at ws or cs. Formal proofs are conducted by writing what is called “proof scores” [3] in CafeOBJ and executing them with CafeOBJ. Proof scores are developed by simultaneous structural induction on a state variable of the OTS. We verify that Anderson enjoys the mutual exclusion property by proving that the property is an invariant of the OTS formalizing Anderson. The verification requires the use of some additional lemmas, one of them is the lemma mentioned above that makes us so stuck to prove its correctness.
The rest of the paper is organized as follows: Sect. II describes Anderson protocol. Sect. III presents how to formally specify the protocol in CafeOBJ. Sect. IV describes our solution to complete the verification by introducing an auxiliary variable and informally discusses why we need to do so. Some related work is mentioned in Sect. VI. Finally, Sect. VII concludes the paper. The specification and proof scores presented in this paper are available at https://gitlab.com/duongtd23/anderson-au.

II. ANDERSON PROTOCOL

We suppose that there are \( N \) processes participating in Anderson protocol. The pseudo-code of Anderson protocol for each process \( p \) can be written as follows:

\[
\text{Loop} \quad \text{“Remainder Section”} \\
rs : place[p] := \text{fetchIncmod}(next, N); \\
ws : \text{repeat until array}[place[p]]; \\
\text{“Critical Section”} \\
cs : \text{array}[place[p]], \\
array[(place[p] + 1) \% N] := \text{false, true};
\]

We suppose that each process is located at \( rs \), \( ws \) or \( cs \) and initially located at \( rs \). \( place \) is an array whose size is \( N \) and each of whose elements stores one from \( \{0, 1, \ldots, N - 1\} \). Initially, each element of \( place \) can be any from \( \{0, 1, \ldots, N - 1\} \) but is 0 in this paper. Although \( place \) is an array, each process \( p \) only uses \( place[p] \) and then we can regard \( place[p] \) as a local variable to each process \( p \). \( array \) is a Boolean array whose size is \( N \). Initially, \( array[0] \) is true and \( array[j] \) is false for any \( j \in \{1, \ldots, N - 1\} \). \( next \) is a natural number variable and initially set to 0. \( \text{fetchIncmod}(next, N) \) atomically does the following: setting \( next \to (next + 1) \% N \) and returning the old value of \( next \). \( x, y := e_1, e_2 \) is a concurrent assignment that is processed as follows: calculating \( e_1 \) and \( e_2 \) independently and setting \( x \) and \( y \) to their values, respectively.

III. FORMAL SPECIFICATION OF ANDERSON PROTOCOL

We use four observation functions \( pc, \) \( next, \) \( place, \) \( array \) to store information about the location of each process, the value of the global variable \( next \), the value stored in each element of \( place \) and the value stored in each element of \( array \), respectively:

\[
op pc : \text{Sys Pid} \to \text{Label} . \\
op next : \text{Sys} \to \text{SNat} . \\
op place : \text{Sys Pid} \to \text{SNat} . \\
op array : \text{Sys SNat} \to \text{Bool} .
\]

\( \text{Sys} \) is the sort that represents the state space of Anderson. \( \text{Pid} \) is the sort denoting the set of process IDs. \( \text{Label} \) is the sort that expresses the set of labels (\( rs, ws \) and \( cs \)). \( \text{SNat} \) is the sort of natural numbers and \( \text{Bool} \) is the sort of Boolean values. Observation function \( \text{array} \) observes the value stored in each element of \( array \) by passing to \( array \) the index of element as the second argument.

We also introduce the observer \( \text{count} \) to keep track of the number of processes that would like to enter the Critical Section and/or to be there (i.e., the number of processes currently located at \( cs \) or \( ws \)):

\[
op \text{count} : \text{Sys} \to \text{SNat} .
\]

\( \text{SNzNat} \) is the sort of non-zero natural numbers and a sub-sort on \( \text{SNat} \). \( N \) is expressed as the constant \( N \) of \( \text{SNzNat} \). The property says that \( N \) is greater than 1 because if there is only one process, we do not need to use any mutual exclusion protocols. In the formal specification, we declare \( 1 \) as a constant of \( \text{SNzNat} \) that equals \( s(0) \) (i.e., \( 1 \) is successor of 0).

We use one constructor that represents an arbitrary initial state as follows:

\[
op \text{init} : \to \text{Sys } \{\text{constr}\} .
\]

\( \text{init} \) is defined in terms of equations, specifying the values observed by the four observation functions in an arbitrary initial state as follows:

\[
eq \text{pc}(\text{init},P) = rs . \quad \text{eq next(\text{init})} = 0 . \quad \text{eq place(\text{init},P)} = 0 . \quad \text{eq count(\text{init})} = 0 . \quad \text{eq array(\text{init},I)} = (\text{if } I = 0 \text{ then true else false fi}) .
\]

where \( P \) is a CafeOBJ variable of \( \text{Pid} \) and \( I \) is a CafeOBJ variable of \( \text{SNat} \).

We use three transition functions that are also constructors:

\[
op \text{want} : \text{Sys Pid} \to \text{Sys } \{\text{constr}\} \\
op \text{try} : \text{Sys Pid} \to \text{Sys } \{\text{constr}\} \\
op \text{exit} : \text{Sys Pid} \to \text{Sys } \{\text{constr}\}.
\]

The three transition functions capture the actions that each process moves to \( ws \) from \( rs \), tries to move to \( cs \) from \( ws \) and moves back to \( rs \) from \( cs \), respectively. The reachable states are composed of the four constructors.

Each of the three transition functions is defined in terms of equations, specifying how the values observed by the four observation functions change. Let \( S \) be a CafeOBJ variable of \( \text{Sys} \), \( P \) & \( Q \) be CafeOBJ variables of \( \text{Pid} \) and \( I \) & \( J \) be CafeOBJ variables of \( \text{SNat} \).

\( \text{want} \) is defined as follows:
ceq pc(want(S,P), Q) = (if P = Q then ws else pc(S, Q) fi)
    if c-want(S, P) .
ceq place(want(S,P), Q) = (if P = Q then next(S) else place(S, Q) fi)
    if c-want(S, P) .
ceq next(want(S,P)) = (s(next(S)) rem N) if c-want(S,P) .
ceq array(want(S,P), I) = array(S, I) .
ceq count(want(S,P)) = s(count(S))
    if c-want(S, P) .
ceq want(S,P) = S if c-want(S, P) = false .

where c-want(S, P) is

pc(S,P) = rs and count(S) < N

s of s(next(S)) is the successor function of natural numbers. x rem y calculates the remainder obtained by dividing x by y. The equations say that if c-want(S, P) is true, the location of P changes to ws, the location of each other process Q does not change, place does not change, next does not change, and array does not change in the state denoted want(S,P); if c-want(S, P) is false, nothing changes.

try is defined as follows:

ceq pc(tray(S,P), Q) = (if P = Q then cs else pc(S, Q) fi)
    if c-try(S, P) .
c eq place(tray(S,P), Q) = place(S, Q) .
c eq array(tray(S,P), I) = array(S, I) .
c eq next(tray(S,P), I) = next(S) .
c eq count(tray(S,P)) = count(S) .
ceq tray(S,P) = S if c-try(S, P) = false .

where c-try(S, P) is

pc(S,P) = ws and array(S,place(S,P)) = true

The equations say that if c-try(S, P) is true, the location of P changes to ws, the location of each other process Q does not change, place, next and count do not change in the state denoted tray(S,P); if c-try(S, P) is false, nothing changes.

exit is defined as follows:

ceq pc(exit(S,P), Q) = (if P = Q then rs else pc(S, Q) fi)
    if c-exit(S, P) .
c eq place(exit(S,P), Q) = place(S, Q) .
c eq array(exit(S,P), I) = array(S, I) .
c eq next(exit(S,P)) = next(S) .
c eq count(exit(S,P)) = (sd(count(S), 1))
    if c-exit(S, P) .
ceq exit(S,P) = S if c-exit(S, P) = false .

where c-exit(S, P) is

pc(S,P) = cs.

The equations say that if c-exit(S, P) is true, the location of P changes to rs, the location of each other process Q does not change, place does not change, next does not change, count is decreased by one, the Ith element of array is set true if I equals s(place(S,P)) rem N, the Jth element of array is set false if J equals place(S,P), and each other element of array does not change in the state denoted exit(S,P); if c-exit(S, P) is false, nothing changes.

IV. FORMAL VERIFICATION BY PROOF SCORES

The mutual exclusion property is specified as follows:

eq mutex(S,P,Q) = ((pc(S,P) = cs and pc(S,Q) = cs) implies (P = Q)) .

The equation says that if there are processes in the critical section, there is one, namely that exists at most one process in the critical section at any given moment.

We prove mutex(S,P,Q) for all reachable states S and all process IDs P & Q by structural induction on S. There are four cases to tackle: (1) init, (2) want, (3) try and (4) exit. Let us consider case (3). What to prove is mutex(tray(s,r),p,q), where s is a fresh constant of Sys representing an arbitrary state and p, q and r are fresh constant of Pid representing arbitrary process IDs. The induction hypothesis is mutex(s,P,Q) for all process IDs P & Q. Let us note that s is shared by mutex(tray(s,r),p,q) and mutex(s,P,Q), while the variables P and Q can be replaced with any terms of Pid, such as p and q.
Figure 1 shows the case splitting strategy to prove case (3). Case (3) is first split into two sub-cases: (3.1) \(pc(S, r) = ws\) and (3.2) \(pc(S, r) = cs\) = false. Case (3.2) can be discharged, its proof score fragment is as follows:

open INV .
op s : -> Sys . ops p q r : -> Pid .
eq (pc(s,r) = ws) = false .
red mutex(s,p,q) implies mutex(try(s,r),p,q) .
close

where INV is the module in which the specification of Anderson together with mutex are available, open makes the module INV available, close stops the use of the module and red reduces (computes) the given term. Feeding this proof score fragment into CafeOBJ, CafeOBJ returns true, meaning that the case is discharged.

Case (3.1) is applied case splitting several more times as shown in the Figure. With case (3.1.1.2.1), it requires us to use a lemma to discharge the sub-case. Case (3.1.1.2.1) says that process \(p\) is located at \(cs\), process \(r\) (or \(q\) since \(q = r\)) is located at \(ws\) and \(array(s,place(s,r)) = true\). In this case, process \(r\) can move to \(cs\), breaking the property concerned because there are two processes \(p\) and \(r\) located at \(cs\). That is the reason why we need to conjecture a lemma to discharge this case. Such a lemma can be conjectured from the assumptions made in this case. We have conjectured inv1 as such a lemma, which is as follows:

\[
eq \text{inv1}(S, P, Q) = \left(\text{pc}(S, P) = ws \text{ and } \text{pc}(S, Q) = cs\right) \Rightarrow (s^0(0) < \text{count}(S)).
\]

Then, in the proof score of case (3.1.1.2.1), we use inv1 as a lemma:

open INV .
op s : -> Sys . ops p q r : -> Pid .
eq \text{pc}(s, r) = ws .
eq \text{array}(s, \text{place}(s, r)) = true .
eq p = r . \text{eq } (q = r) = false .
eq \text{pc}(s, q) = cs .
red inv1(s, r, q) implies mutex(s, p, q) implies mutex(try(s, r), p, q) .
close

The remaining cases can be discharged likewise. The proof of mutex does not require any other lemma except for inv1. We need to prove that inv1 is an invariant of the OTS formalizing Anderson to complete the verification. If inv1 is not an invariant of the OTS formalizing Anderson, there exists a state such that there are two different processes \(P\) and \(Q\) where \(P\) can freely enter the critical section and \(Q\) is locating at the critical section or can freely enter the critical section. In both cases, the mutual exclusion property is broken. That is the reason why inv1 must be an invariant of the OTS formalizing Anderson.

In the proof of inv1, we need to use another lemma inv7 that is as follows:

\[
eq \text{inv7}(S, P) = \left(\text{pc}(S, P) = ws \text{ or } \text{pc}(S, P) = cs\right) \Rightarrow (0 < \text{count}(S)).
\]

where \(S\) is a CafeOBJ variable of \(Sys\), \(P\) & \(Q\) are CafeOBJ variables of \(Pid\). inv7 intuitively says that if there exists a process located at \(ws\) or \(cs\) in a state \(S\), \(\text{count}(S)\) is greater than 0. Considering the roles of \(\text{count}\) that keeps track of the number of processes that have moved to \(ws\) and not yet left \(cs\), namely the number of processes that are located at \(ws\) or \(cs\), inv7 must be an invariant of the OTS formalizing Anderson.

In the proof of inv7, we need to use a lemma inv7-2 that is as follows:

\[
eq \text{inv7-2}(S, P, Q) = \left(\text{pc}(S, P) = ws \text{ or } \text{pc}(S, P) = cs\right) \Rightarrow \left(\text{pc}(S, Q) = ws \text{ or } \text{pc}(S, Q) = cs\right) \Rightarrow (P = Q) = false \Rightarrow (s(0) < \text{count}(S)).
\]

inv7-2 intuitively says that if there are two different processes located at \(ws\) or \(cs\) in a state \(S\), \(\text{count}(S)\) is greater than 1.

In the proof of inv7-2, we need to use a lemma inv7-3 that is as follows:

\[
eq \text{inv7-3}(S, P, Q, R) = \left(\text{pc}(S, P) = ws \text{ or } \text{pc}(S, P) = cs\right) \Rightarrow \left(\text{pc}(S, Q) = ws \text{ or } \text{pc}(S, Q) = cs\right) \Rightarrow \left(\text{pc}(S, R) = ws \text{ or } \text{pc}(S, R) = cs\right) \Rightarrow (P = Q) = false \Rightarrow (P = R) = false \Rightarrow (Q = R) = false \Rightarrow \left(s^0(0) < \text{count}(S)\right).
\]

where \(R\) is a CafeOBJ variable of \(Pid\). inv7-3 intuitively says that if there are three different processes located at \(ws\) or \(cs\) in a state \(S\), \(\text{count}(S)\) is greater than 2.

It seems necessary to use an unlimited number of similar lemmas to complete the proof of inv7. If CafeOBJ made it possible to use an operator with a variable number of parameters, we could generalize the lemmas:

\[
eq \text{inv7-k}(S, P, P_1, ..., P_k) = \left(\text{pc}(S, P_1) = ws \text{ or } \text{pc}(S, P_1) = cs\right) \Rightarrow \left(\text{pc}(S, P_2) = ws \text{ or } \text{pc}(S, P_2) = cs\right) \Rightarrow \left(\text{pc}(S, P_k) = ws \text{ or } \text{pc}(S, P_k) = cs\right) \Rightarrow (P_1 = P_2) = false \Rightarrow (P(k-1) = P_k) = false \Rightarrow (s^0(k-1)(0) < \text{count}(S)) .
\]

where \(s^0(k-1)(0)\) denotes \(k-1\). \(k\) can vary like a variable and the number \(k\) of parameters can change. It is, however, impossible to deal with such an operator with a variable number of parameters in CafeOBJ. That is the reason why we got stuck several months to prove that Anderson enjoys the mutual exclusion property. If we could complete the proof of inv7, then the verification is accomplished.

V. INTRODUCING AN AUXILIARY VARIABLE

A. Introducing psInWsCs

The proof of inv7 seems so tough despite its obviousness. We overcome the problem by introducing an observer (also can be called an auxiliary variable) psInWsCs that records
where \( P \) denotes the set of processes currently located at \( \text{ws} \) or \( \text{cs} \). The observer is declared as follows:

\[
\text{op psInWsCs : Sys -> SetPids .}
\]

where \( \text{SetPids} \) is the sort denoting the set of process IDs. \( \text{psInWsCs} \) is defined in the initial states and each transition as follows:

\[
\text{eq psInWsCs(init) = emp .}
\]

\[
\text{ceq psInWsCs(want(S,P)) = c\text{-want}(S,P) .}
\]

\[
\text{eq psInWsCs(try(S,P)) = psInWsCs(S) .}
\]

\[
\text{ceq psInWsCs(exit(S,P)) = c\text{-exit}(S,P) .}
\]

where \( \text{emp} \) is the constant of sort \( \text{SetPids} \) representing the empty set. The equations say that initially, \( \text{psInWsCs} \) is empty; when process \( P \) moves to \( \text{ws} \) from \( \text{rs} \), \( P \) is inserted into \( \text{psInWsCs} \); when process \( P \) moves to \( \text{rs} \) from \( \text{cs} \), it is removed from \( \text{psInWsCs} \); \( \text{psInWsCs} \) does not change when \( P \) moves to \( \text{cs} \) from \( \text{ws} \).

Then, we can complete the proof of \( \text{inv7} \) by using the following lemmas:

\[
\text{eq inv10(S) = \#(psInWsCs(S)) .}
\]

\[
\text{eq inv11(S,P) = (pc(S,P) = \text{ws} or pc(S,P) = \text{cs}) implies P \ \text{\in psInWsCs}(S) .}
\]

\[
\text{eq inv12(SE,E) = E \ \\text{\in SE implies 0 < \#(SE) .}}
\]

where \( \# \) is the operator taking a set as the parameter and returning the size of it, \( \text{\in} \) is the infix operator checking the existence of an element in a set, and \( \text{SE} \) and \( \text{E} \) are CafeOBJ variables denoting arbitrary sets and elements, respectively. To prove \( \text{inv7} \), we do not need to apply structural induction, but only case splitting is enough. Table I shows the case splitting for the proof of \( \text{inv7} \), where each \( \text{cbsi} \) for \( i = 1, 2, 3 \) is as follows:

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>( pc(s,p) = \text{cs} ), ( \text{cbs1} ), ( \text{cbs2} ), ( \text{cbs3} )</td>
</tr>
<tr>
<td>1.1.2</td>
<td>( pc(s,p) = \text{cs} ), ( \text{cbs1} ), ( \text{cbs2} ), ( \neg \text{cbs3} )</td>
</tr>
<tr>
<td>1.2</td>
<td>( pc(s,p) = \text{cs} ), ( \text{cbs1} ), ( \neg \text{cbs2} )</td>
</tr>
<tr>
<td>2</td>
<td>( pc(s,p) = \text{rs} )</td>
</tr>
<tr>
<td>3.1.1.1</td>
<td>( pc(s,p) = \text{ws} ), ( \text{cbs1} ), ( \text{cbs2} ), ( \text{cbs3} )</td>
</tr>
<tr>
<td>3.1.2</td>
<td>( pc(s,p) = \text{ws} ), ( \text{cbs1} ), ( \text{cbs2} ), ( \neg \text{cbs3} )</td>
</tr>
<tr>
<td>3.2</td>
<td>( pc(s,p) = \text{ws} ), ( \text{cbs1} ), ( \neg \text{cbs2} )</td>
</tr>
</tbody>
</table>

For example, the proof fragment of case (1.1.2) is as follows:

\[
\text{open INV .}
\]

\[
\text{ops p r : -> Pid . op s : -> Sys .}
\]

\[
\text{eq pc(s,p) = \text{cs} .}
\]

\[
\text{eq count(s) = \#(psInWsCs(s)) .}
\]

\[
\text{eq p \ \text{\in psInWsCs}(s) = true .}
\]

\[
\text{eq (0 < \#(psInWsCs(s))) = false .}
\]

\[
\text{red inv12(psInWsCs(s),p) implies inv7(s,p) .}
\]

The proof of this case uses \( \text{inv12} \) as a lemma. Let us repeat again that, to prove \( \text{inv7} \), we do not apply structural induction, but only case splitting. The proof of case (3.1.1.2) also uses \( \text{inv12} \) as a lemma. The proofs of cases (1.1.2) and (3.1.2) use \( \text{inv11} \) as a lemma. The proofs of cases (1.2) and (3.2) use \( \text{inv10} \) as a lemma. The remaining cases are proved without any lemma.

To complete the verification, we also use the following lemmas:

\[
\text{eq inv2(S,P) = ((pc(S,P) = \text{cs}) implies (array(S,place(S,P)) = true)) .}
\]

\[
\text{eq inv3(S,G,H) = ((G = \text{H}) = false and array(S,G) = true) .}
\]

\[
\text{eq inv4(S,P,Q) = (place(S,P) = place(S,Q) and (pc(S,P) = \text{rs}) = false and (P = Q) = false) implies (pc(S,Q) = \text{rs}) .}
\]

\[
\text{eq inv5(S,I) = (pc(S,P) = \text{ws and place}(S,P) = ((I + count(S)) \text{ rem No})) implies ((I + count(S)) < No) = false .}
\]

\[
\text{eq inv6(S,P) = (place(S,P) < No) .}
\]

\[
\text{eq inv8(S,I) = (array(S,I) = true implies next(S) = (I + count(S)) \text{ rem No}) .}
\]

\[
\text{eq inv9(S) = (next(S) < No) .}
\]

The proof of \( \text{inv1} \) uses \( \text{mutex} \), \( \text{inv4} \), \( \text{inv8} \), and \( \text{inv7} \) as lemmas. The proof of \( \text{inv2} \) uses \( \text{mutex} \) as a lemma. The proof of \( \text{inv3} \) requires the use of \( \text{inv2} \) as a lemma. \( \text{inv4} \) would be the most complicated invariant, its proof uses \( \text{inv1, inv2, inv5, inv6, inv7, and inv8} \) as lemmas. To prove \( \text{inv5} \), we need to use \( \text{inv2, inv4, inv6, inv7, and inv8} \) as lemmas. \( \text{mutex} \) and \( \text{inv9} \) are used as a lemma in the proof of \( \text{inv6} \). The proof of \( \text{inv8} \) uses \( \text{inv2} \) and \( \text{inv3} \) as lemmas. We can prove \( \text{inv9, inv11, and inv12} \) without using any other lemma. The proof of \( \text{inv10} \) requires to use \( \text{inv11} \) as a lemma.

B. Discussion

To prove a property is an invariant of an OTS, we need to conjecture some additional lemmas that are also invariants on the fly during the proof. It is often the case such that the lemma conjectured is not easy to prove such as \( \text{inv7} \) in this paper. Sometimes, in some non-trivial sub-cases of the induction proof, we do not have enough information to verify that the lemma is preserved by a transition. Let us return to \( \text{inv7} \) and its lemmas \( \text{inv7-2, inv7-3} \), etc. in the last section to explain the difficulty made us could not complete the proofs of them and the reason why it is necessary to introduce auxiliary variables like \( \text{psInWsCs} \).
Anderson, which is called A-Anderson protocol, and for-

The premise of inv7 says that there exists a set of processes that currently located at ws or cs, and its corresponding conclusion concludes that count is greater than or equal to the size of that set. However, we do not have enough information to calculate the value of count to make the comparison because we can not observe the full set of all processes currently located at ws or cs. We only know that there explicitly is/are one (P), or two (P and Q), or three (P, Q, and R) process(es) currently located at ws or cs corresponding to each inv7, or inv7-2, or inv7-3, respectively. That is the reason why the proof of inv7 or each of inv7-k becomes so tough or even almost impossible. Since the difficulty comes from the impossibility of observing the full set of all processes currently located at ws or cs, we introduce the observer psInWsCs recording the collection of processes that have entered ws and not yet left cs. Consequently, we overcome the difficulty, accomplish the proof inv7 as well as the complete verification.

VI. RELATED WORK

Tran and Ogata [2] have made an abstract version of Anderson, which is called A-Anderson protocol, and formally verified that A-Anderson enjoys the mutual exclusion property. The verification is conducted in three ways: (1) by writing proof scores in CafeOBJ, (2) with a proof assistant CiMPA [5] for CafeOBJ and (3) with a proof generator CiMPG [5] for CafeOBJ. The paper has also mentioned how to formally verify that Anderson enjoys the mutual exclusion property by showing that there exists a simulation relation from Anderson to A-Anderson, and such simulation preserves the property. The details of this verification technique, however, were not presented in [2] due to the page-limitation. They mentioned that they would report that part in a longer version.

Lamport and Merz [6] has described how to introduce auxiliary variables into TLA+ specifications to prove a refinement mapping between two TLA+ specifications (i.e., the set of observable behaviors of the first specification is a subset of the behaviors of the second one). Auxiliary variables have been classified into three kinds: history, prophecy and stuttering variables. History variables are used to record what has happened in the past (including the present). Prophecy variables are used to predict what will happen in the future. Stuttering variables are used to introduce stuttering steps. psInWsCs we have used in this paper corresponds to a history variable. Lamport and Merz use auxiliary variables to make it possible to find a refinement map from a TLA+ specification to another TLA+ specification. While we use psInWsCs to complete the proof that a property is an invariant of the OTS formalizing Anderson.

Auxiliary variables, go back to the past were originally introduced by Owicki and Gries [7] in the form of history variables. Later, Abadi and Lamport [8] have introduced the idea of prophecy variables. In [8], Abadi and Lamport have presented how to use both history and prophecy variables to prove that one program is a correct implementation of a specification, by showing that the former refines the latter.

VII. CONCLUSION

We have formally verified that Anderson protocol to which an auxiliary variable is introduced enjoys the mutual exclusion property. Consequently, we can conclude that Anderson enjoys the property. Originally, we got stuck several months in the verification attempt because the proof requires a lemma that is so tough to prove the correctness of it. Introducing an auxiliary variable psInWsCs helps us to accomplish the proof of that lemma, leading to the complete formal verification. psInWsCs records all processes currently located at cs or ws, which means that it does not affect the current or future values of any other variables. Thus, it can be guaranteed that adding psInWsCs does not change the behavior of Anderson.

Conjecture lemma has been considering as one of the most challenging tasks to formally prove that a property is an invariant of an OTS. Normally, we can not always conjecture the best lemma every time we need to use a lemma. Sometimes, the lemma is so tough or even almost impossible to prove such as inv7 in this paper. Then, introducing auxiliary variables into the specification can help us to complete the lemma’s proof as well as the formal verification such as psInWsCs in the present paper. We can understand the reason why we need to introduce psInWsCs in the formal verification of Anderson case study. However, in general, we have not had a contended answer for the question: when we need to introduce auxiliary variables to complete formal verification of other case studies? That should be one piece of our future work to answer such a question.

REFERENCES

Abstract—Quality assurance of rapidly evolving systems is increasingly important for their deployment to real-life applications. Despite the challenges posed by the increasing complexity of these systems, various techniques have been developed to check their correctness, such as theorem proving, which is a powerful formal verification method that can provide a complete guarantee. However, the proving process in the interactive theorem provers like Coq highly relies on human interactions, making the proving process difficult and time-consuming. To automate the proving process in Coq, we present a framework for predicting tactics in Coq by using Long Short Term Memory (LSTM). We take into account the effect of the dataset proof style on machine learning and create a new dataset following a specific proof style. We use the generated data to train an LSTM-based neural network that could give tactic predictions based on the proof context. This neural network reaches an accuracy of 58% if we only use the first predicted tactic and reaches an accuracy of 87% if we select the first three tactic suggestions, achieving a 15.2% and 12.8% improvement rate, respectively, compared to the methods in previous work.

I. INTRODUCTION

In the last few decades, our reliance on software systems has rapidly grown. Quality assurance of such systems is thus necessary and crucial for their deployment to real-life applications. Despite the challenges posed by the increasing complexity of these systems, many techniques have been developed to check their functional correctness, e.g., testing and verification. Compared with testing techniques, formal verification could provide a complete guarantee of the critical properties of software systems. Theorem proving is one of the most popular formal verification methods, where systems are modeled in an appropriate mathematical logic, and critical properties are represented as propositions to be proved and verified in theorem provers. Up to present, theorem proving has been successfully applied to various domains such as computer science [1], artificial intelligence [2], economy [3], biomedical [4] and self-adaptive systems [5].

Theorem provers are mainly categorized into two types: Automated Theorem Provers (ATPs) such as Alt-Ergo [6], and SPASS[7], and Interactive Theorem Provers (ITPs) such as Coq [8], Isabella [9], and PVS [10]. Though the reasoning process is automated in ATPs, they usually suffer from complexity and expressive power problems. In contrast, the expressive power of ITPs are usually stronger, which makes them more suitable for the formalization of “most non-trivial theorems in mathematics or computer system correctness” [11]. However, they require human interaction with computer in the process of proof construction, which is the reason that they are also called proof assistants. Coq is one of the interactive theorem provers. It allows users to declare propositions and prove them. When users are proving propositions, they are actually constructing proof terms with the help of commands called tactics. This process of constructing proofs highly relies on human interactions, which can make the proving process difficult and time-consuming. It is quite often that some intuitively obvious facts require tedious proofs in Coq. In general, proving process in most ITPs is labor-intensive due to the lack of automation.

Many efforts have been made to improve the degree of automation of interactive theorem provers. For example, the Coq team provided Ltac [12] to support custom tactics and a set of automatic tactics like auto and congruence that realize partial automation in certain domains. Recently machine learning techniques have been investigated to automate the proving process. [13] presented ML4PG that gathers data from a general proof interface ProofGeneral, and used clustering algorithms to learn and predict dependencies of goals. [14] tried several machine learning techniques to learn proof dependencies from formalization done with Coq system. TacticToe was developed in [15] for HOL4 theorem prover, which implements a modified A* algorithm to automate the tactic selection in proof search. [16] presented a reinforcement learning environment for theorem proving and a deep learning driven automated theorem prover for higher-order logic. [17] leveraged Recurrent Neural Network (RNN) to predict Coq tactics for property verification in the domain of coordination language. [18] used k-nearest neighbor algorithm to learn from previous proof scripts for tactic proof search in Coq. [16], [17], and [18] all work on tactic level learning, through which the predicted tactics can be directly applied to proceed the proof and custom tactics can also be supported. However, the focus of most existing works has been more on the modeling design than on the dataset construction, where the datasets are mostly directly extracted from raw proof scripts such as standard Coq libraries. We need to put more efforts on the construction of dataset to obtain higher quality data so that, combined with an effective learning model, we can achieve better automating performance.

In this paper we present a framework for predicting tactics in Coq using a Long Short Term Memory-based (LSTM-based) neural network [19]. We re-prove a theorem library following a specific proving style to create a new dataset, and use this dataset to train an LSTM neural network to learn and predict potential tactics with several hypotheses and a proof-goal as input. The contributions of our work can be summarized as follows:

1) We take a different approach to create training and test data. The specific proving style we adopted to re-prove...
theorems makes the learning task more consistent and simple for LSTM-based networks.

2) We design an LSTM-based neural network to learn and predict tactics in Coq. This new architecture could stabilize hidden state dynamics and reduce overfitting, resulting in better performance for automating the proving process.

3) We perform experimental evaluation on the effectiveness of our method, which reaches an accuracy of 58% if we only use the first predicted tactic, and reaches an accuracy of 87% if we use the top three tactic suggestions, achieving a 15.2% and 12.8% improvement rate, respectively, compared to the baseline.

The rest of this paper is organized as follows: In Section II we briefly introduce how tactics work in Coq. The construction of our dataset and pre-processing steps are explained in Section III. The design of our LSTM-based neural network is elaborated in Section IV, and evaluation of our approach is provided in Section V. Finally, Section VI concludes this paper and discusses some future research directions.

II. BACKGROUND

A. Tactics in Coq

As an interactive theorem prover, Coq allows users to declare and prove propositions and then to extract certified programs from the certified proofs. After declaring a proposition in Coq, users enter the proof mode. In this mode, the proposition to be proved is called a goal, and users can apply commands called tactics to decompose the goal into simpler subgoals or to solve it directly. The decomposition process ends when there are no more subgoals. An interaction in Coq is a context-tactic pair as shown in Figure 1, where the context contains hypotheses we currently have and a set of subgoals we need to prove.

**Fig. 1. An Interaction in Coq**

Some tactics can be directly applied to the goal, others require arguments. For example, the split tactic can be directly applied to a conjunctive goal with no arguments, while the apply tactic means applying a known theorem to the goal and thus requires arguments, and the intros tactic can be used either with or without arguments, only differing on the names of the introduced hypotheses. In our framework, we only consider predicting tactic names, with tactic arguments excluded, since higher order logic is undecidable.

Different tactics may have the same effect on some goals. For example, when there is only one hypothesis to be introduced, the effects of the intro tactic and the intros tactic are the same. Besides, in most cases there is more than one way to construct the proof, that is to say, a proposition can be proved by different sequences of tactics, different users may have different proof styles of using tactics.

B. Assumptions

Several assumptions on proving process in Coq are made in [17] based on the observations and expert experience, which turns out to be effective in tactic prediction. We follow the assumptions (1-3) made in [17] in our framework. Besides that, we make an extra assumption (4). Basically, the first two assumptions allow us to use only hypotheses and the first subgoal to make predictions without considering all the subgoals. The third and the forth assumptions serve as the basis for data pre-processing and data augmentation respectively, which we will describe in more details later on. All the assumptions are summarized as follows:

1) Proofs of subgoals are independent of each other, which means that tactics to prove a subgoal do not depend on proofs of any other subgoals.
2) Rearranging the order of subgoals is not considered in our proving process, so the tactic we use and suggest is only applied to the first subgoal or current hypotheses.
3) When looking for a proper tactic to use, the structure of a Coq term is more important than its content.
4) Tactics can be applied to either a subgoal or one or more hypotheses.

III. DATASET AND DATA PRE-PROCESSING

A. Dataset Construction

Using a neural network to predict appropriate tactics for proving process requires a dataset for training and testing. As we mentioned in the previous section, proofs in Coq can be written in different proof styles. There are good and bad proof styles for neural network learning. Although it is difficult to formally specify what a good proof style is, we provide several heuristics to distinguish a good proof style from a bad one. A good proof style for learning is supposed to be consistent, otherwise it can cause confusion for the learning process. For example, if we use different tactics in similar context, in other words, we follow an inconsistent proof style, then the minimization of loss function will be hindered, resulting in poor prediction performance. Besides, we should also consider the inherent learning difficulty of the proof style. A good proof style should not use overly complicated tactic mechanism, nor should it have tactics that are seldom used, so that it would be more simplified and consistent for a machine learning model to learn.
The standard Coq libraries constitute a large dataset for machine learning, many approaches are developed based on this dataset. However, according to the heuristics these theorem libraries’ proof styles are not suitable for neural network learning for the following reasons:

1) Inconsistency: Since these libraries are developed by different authors, they usually have different proof styles. For example, SSReflect is a collection of libraries for the SSReflect [20] proof language and its proof style is quite different from the others.

2) Complexity: Advanced tactics are intensively used for conciseness and robustness, such as using tactical to combine several tactics as a compound tactic, which makes it more difficult to learn.

3) Infrequently used tactics: Some libraries define custom tactics to reduce repetition, but these tactics will neither be used in other libraries nor in practical proving process. Therefore, we cannot utilize the standard Coq libraries as our dataset, instead we need to create a new dataset with a specific proof style that is not only consistent but also easy to learn.

Following the above heuristics, we manually create a theorem library about the properties of Reo [21] connectors in the domain of coordination language as our dataset. This domain-specific theorem library is constructed based on the rough proof scripts provided in [17]. The proof style we use when building the theorem library is quite like that of a novice, so we call this proof style the novice proof style, the benefits of which are summarized as follows:

1) Consistency. All proofs are written in a consistent style, we prioritize different tactics, so that for similar context we always use the same tactic to proceed the proof. 

2) Simplicity. We only use one tactic at a time, and we avoid using unnecessary repeated tactics, for example, we use a single intros to introduce all the hypotheses and variables instead of a series of intros (Note that intro and intros are two different tactics with similar names and functions).

3) Restricted tactics. We restrict ourselves to a set of frequently used tactics to write proofs, including 23 tactics in total, as illustrated in Figure 2.

When building our theorem library, we add a few new lemmas to complete the original proof. In the end, the constructed dataset contains 31 theorems and lemmas, all of which are fully proved, with a total of 830 lines of codes, while the original one contains 1 fully proved theorem and 9 partially proved theorems, with a total of 383 lines of codes.

**B. Data Pre-Processing**

We follow the pre-processing method in [17], which contains three steps as follows:

1) Use a Python script to write the Coq proof scripts line by line to SerAPI [22], which performs machine-to-machine interaction with Coq through S-expressions and extracts context-tactic pairs.

2) Refactor extracted Coq terms by adding corresponding term types as structural information based on the assumption (3).

3) Perform word encoding on hypotheses and the goal to get fixed-length vectors, since Coq programming language does not have a finite dictionary nor semantics similarity. We fix a maximal length of words, and for each word we map its character to its ASCII code and fill the rest part with zero.

4) Apply one-hot encoding on tactic names to get one-hot vectors as sample labels.

However, we use different parameters for word encoding to eliminate redundant zeros. The purpose of this adjustment is to reduce input dimensions (dimensions of input are reduced from 5120 to 1360) and computational complexity. Since we use a restricted set of tactics, the dimension of the output is also reduced.

In the end, we obtain 526 samples from the constructed theorem library. These samples constitute a database in our learning framework, each of which is composed of a variable length sequence as input and a one-hot vector as output. The sequence consists of the first subgoal and several hypotheses if there is any, and the subgoal is always the last item in the sequence. The dataset in [17] contains 173 samples, but there are 12 samples labeled by the ‘admit’ tactic, which cannot be used in a full proof, thus reducing the actual samples from 173 to 161.

**IV. Learning Tactics Through LSTM**

According to the assumptions, our problem can be regarded as a sequence classification problem, for which the most popular solution is RNNs. However, vanilla RNNs suffer from issues of vanishing gradient. To reduce impacts of the gradient vanish problem, LSTM (with cell memory and gate control) [19] is proposed and widely adopted to deal with tasks when long-term dependencies need to be captured. Therefore, we choose to build an LSTM-based neural network in our framework. The structure of our neural network is shown in Figure 3, including an LSTM layer, a layer normalization layer, a dropout layer and a fully-connected layer.

- LSTM Layer: The neural network has an LSTM layer containing 512 self-connected hidden units. The activation function is \( \tanh \) and the gate activation function is Sigmoid function.
- Layer Normalization Layer: After the LSTM layer we add a layer normalization layer where the output of LSTM layer is normalized to zero mean and unit variance.
- Dropout Layer: Cells in this layer are randomly disconnected according to dropout rate (set as 0.5 in our framework) when training.

![Supported Tactics in Our Framework](image)
• Fully-connected Layer: A fully-connected layer uses $\text{softmax}$ as its activation function to normalize its output as a probability distribution.

The loss function of our neural network is cross-entropy function, which measures the difference between two probability distributions. For two probability distributions $p$ and $q$, their cross-entropy is defined as:

$$H(p, q) = - \sum_x p(x) \cdot \log q(x)$$

The layer normalization [23] can stabilize hidden state dynamics for recurrent neural network and also help with reducing training time. The normalization is not implemented in unit-level but in layer-level, because the unit-level approach is much more complicated and more computational expensive but has roughly the same performance as layer-level approach.

In order to overcome the overfitting problem caused by the limited data and high input dimensions, we add the dropout layer. We also use label smoothing [24] technique, which is another frequently used regularization method. With label smoothing, the hard 0 and 1 classification targets in the ground truth one-hot vector $y$ will be replaced with targets of $\frac{\varepsilon}{k-1}$ and $1 - \varepsilon$ respectively, where $\varepsilon$ is the smoothing parameter and $k$ is the class number, thus prevents the pursuit of hard probabilities without discouraging correct classification.

We also use data augmentation to generate more training data to deal with the overfitting problem. In the pre-processing step, the context in an interaction is transformed into a sequence, where the last item is the goal to be proved and the others are hypotheses. Users who are familiar with Coq should be aware that the order of hypotheses is independent of the tactic that can be used. This fact inspires us to perform data augmentation by shuffling hypotheses. We perform shuffling on the goal and hypotheses together, since tactics can be applied to either a subgoal or any hypotheses according to assumption (4), which means that there is no essential difference between the goal and hypotheses for predicting tactics. Each sample sequence in the training set is shuffled 440 times in a way that every synthetic sample is different from other samples whenever possible. For those samples with too few hypotheses to get enough distinct synthetic samples (less than 5 hypotheses since $5! < 440 \leq 6!$), we use their full permutations as the generated samples.

In the training process, the neural network is trained for 30 epochs with a batch size of 256, and we use the RMSprop optimizer with learning rate set as 0.001, $\rho$ set as 0.9, $\varepsilon$ set as $10^{-7}$ and clipnorm set as 0.8. The label smoothing is 0.1. Initial kernel weights of LSTM layer are set by Glorot uniform initializer, initial recurrent weights of LSTM layer are set orthogonally, and initial bias is set as zero vector.

V. Evaluation

Our neural network is implemented in Keras [25], a high-level neural networks API in Python. Multiple popular machine learning frameworks are supported by Keras, and we use Tensorflow as its backend. We train the neural network on our dataset and [17]’s dataset separately. Experiments are run
Fig. 5. The prediction for an validation sample and its true label

with an NVIDIA Tesla P100 GPU, 2 cores of Intel Xeon as CPU and 13 GB memory.

As there are often multiple tactics that can help with the proof process, the evaluation should not be constrained by a single correct answer provided by the proof scripts. Instead, we use \textit{n-correctness rate} \cite{17} to evaluate the tactic prediction performance. \textit{n-correctness rate} measures the likelihood that the top \( n \) tactics predicted by the network are actually useful for theorem proving, which is defined as follows.

\textbf{Definition 5.1 (n-correctness rate):} The output of the network is a probability distribution, if the probability of the targeted tactic (provided by the dataset) is in top \( n \), we say that this prediction is \( n \)-correct, and the corresponding correctness rate is called \( n \)-correctness rate.

We use cross validation to evaluate the tactic prediction performance of our approach, with the performance of the method in \cite{17} as comparison baseline. We perform ablation experiments to evaluate the effectiveness of the constructed dataset and network design. Specifically, the original neural network is trained on our new dataset to evaluate the usefulness of the dataset. The proposed neural network is trained on the original dataset to evaluate the effectiveness of the network design. On our dataset, we evaluate neural network performance by repeating 10-fold cross validation 10 times. In other words, 10-fold cross validation procedure is repeated 10 times and the mean result across all runs is regarded as our final evaluation. While on the original dataset, we use leave-one-out cross validation to evaluate the performance, given that this dataset is smaller. Data augmentation is applied when training our model, but each sample in the original training set is shuffled 130 times, not 440 times. We use this evaluation method due to the limited size of the datasets, where the train-test split method may result in different distributions of training and test sets. The experiment results are shown in Figure 4.

Compared to baseline, our approach achieves a 15.2\%, 17.8\%, and 12.8\% improvement rate on the 1-correctness, 2-correctness, and 3-correctness rate. With regard to the effectiveness of the proposed network design, on the original dataset the 1-correctness, 2-correctness, and 3-correctness rate of our neural network is 0.62\%, 0.62\%, and 1.86\% higher than the original neural network, respectively. On our dataset, the improvements are 1.77\%, 1.98\%, and 0.61\%, respectively. Regarding the usefulness of the constructed dataset, the 1-correctness, 2-correctness, and 3-correctness rate of our neural network on our dataset is improved by 11.7\%, 17.4\%, and 12.0\%, respectively, compared to the original dataset. As for our neural network, the improvement rates are 13.8\%, 16.7\%, 10.2\%, respectively.

The design of our neural network, including the architecture and the use of regularization techniques, has led to improved performance. But the constructed dataset serves as a major factor contributing to the performance improvement, which significantly reduces the machine learning difficulty. In summary, our LSTM-based neural network outperforms the original neural network on both datasets, and our dataset is
easier for LSTM-based neural network to learn, both of which together make our approach perform better.

After taking a closer look at how our neural network performs on the validation set, we find an interesting phenomenon. On some samples, our neural network gives different predictions from the ground truth. However, these different predictions actually can solve the goal, which is exactly the situation we mentioned before. Figure 5 is an illustration for such phenomenon, where both the tactic we use and the prediction of our network can solve the goal. This interesting phenomenon indicates that our n-correctness-based evaluation of our model performance is very pessimistic, the actual performance should be even better.

VI. CONCLUSION

In this paper, we present a framework for predicting tactics to automate the process of proving properties of a specific domain in Coq. In order to automate the proving process in Coq, we create a new dataset by re-proving a theorem library used in [17] in the novice proof style, and train an LSTM-based neural network on this dataset to predict tactics based on proof context. Experiment results show that our approach to creating dataset makes the learning task easier for LSTM-based networks and that the proposed neural network outperforms the baseline, where the correctness of our network is almost 90% if we select first three suggested tactics. Besides, we find that our model is capable of giving suggestions which differ from ground truth in our dataset but can actually proceed the proof. This phenomenon indicates that the actual performance of the network is even better.

In the future, we plan to try reinforcement learning on this problem and try more loss function design for higher correctness rate. Gathering more data is another future work to improve network performance. Since manually building proofs is inefficient, we can try generating simple proofs from existing complex proofs.

ACKNOWLEDGMENTS

This research was supported in part by the Guangdong Science and Technology Department (Grant No.2018B010107004); the National Natural Science Foundation of China under grant No.61772038, 61532019.

References

Formal specification and model checking of a recoverable wait-free version of MCS

Duong Dinh Tran, Kentaro Waki, and Kazuhiro Ogata
School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan
Email: {duongtd,kentaro.waki,ogata}@jaist.ac.jp

Abstract—MCS is widely known as one of the most efficient and influential spinning lock mutual exclusion protocols. The protocol, however, only works under the assumption that processes do not crash while acquiring/releasing the lock or being in the critical section. Furthermore, the exit segment pseudo-code of MCS’s algorithm is not wait-free since a process releasing the lock needs to wait for the next process in the virtual queue to perform some steps. A new version of MCS has been proposed by S. Dhoked and N. Mittal such that the new version is wait-free and recoverable (i.e., if some processes crash, the protocol can recover and work normally). In this paper, we formally specify the recoverable wait-free version of MCS and conduct model checking to check whether the protocol enjoys the mutual exclusion property. Our experiments say that: (1) the property is not satisfied if crashes are allowed to occur without any restriction, (2) the protocol enjoys the property if crashes never happen at all, or (3) if crashes have not occurred recently. We also describe the challenge of how to formally specify dynamic memory allocation and present our solution to solve that problem.

Keywords—mutual exclusion; MCS protocol; wait-free algorithm; recoverable; dynamic allocation

I. INTRODUCTION

Concurrent or distributed systems require efficient mechanisms to handle conflict between concurrent accesses to resources shared among several processes. Mutual exclusion locks are known as one of the most common techniques to solve such problems. Mutual exclusion guarantees that a process only can access the shared resources inside the critical section, and at most one process is allowed to enter the critical section at any time. In 1991, J. M. Mellor-Crummey and M. L. Scott proposed MCS mutual exclusion protocol [1]. Since then, MCS itself together with several variants of it were implemented and used in various environments. For example, variants of MCS have been used in Java Virtual Machines. The numerous implementations and extensive uses imply that the protocol is one of the most efficient and influential mutual exclusion algorithms.

However, a drawback of the MCS algorithm is that its exit segment pseudo-code is not wait-free. An algorithm is wait-free if every action of it by a process completes within a bound number of steps regardless of the behavior of other processes. In MCS, when a process leaving the critical section and releasing the lock, it needs to wait for the next process in the virtual queue to perform some steps. MCS also only works under the assumption that processes do not crash while acquiring/releasing the lock or while in the critical section. Failures or crashes, however, are often possible to happen in real systems. In [2], R. Dvir and G. Taubenfeld have proposed an extension to the original MCS to make the exit segment code wait-free. Based on the augmented MCS in [2], S. Dhoked and N. Mittal [3] have continuously proposed another new version such that it is recoverable and wait-free. The recoverable property says that a process may crash at any point during its execution, but the protocol is able to recover and work as normal. Hereinafter, let us call the recoverable & wait-free version of MCS as RWfMCS.

In this paper, we formal specify RWfMCS and conduct model checking to confirm that when the protocol enjoys the mutual exclusion property and when it does not. Our experiments say that: (1) the protocol does not satisfy the mutual exclusion property if there is not any restriction to the occurrence of crashes; (2) the protocol enjoys the mutual exclusion property if crashes never happen; and (3) the protocol satisfies the mutual exclusion property if crashes have not occurred recently.

Formal specification of RWfMCS has a challenge in specifying dynamic memory allocation. When a process wants to enter the critical section, it first requests for allocating memory to initialize an empty node. Roughly speaking, nodes associated with processes are dynamically created. Unfortunately, modeling dynamic allocation in particular or dynamic systems in general is a non-trivial problem in formal method. P. C. Attie and N. A. Lynch [4] have addressed this problem and presented dynamic I/O automata, which is an extension of I/O automata to model and analyze dynamic systems. In this paper, we overcome that problem by providing a fixed list of “empty nodes” from the beginning. Every time a process requests for allocating memory to construct a new empty node, a top node of the list is extracted and used.

This research was partially supported by JSPS KAKENHI Grant Number JP19H04082.
DOI reference number: 10.18293/SEKE2021-065
II. Preliminaries

A Kripke structure $K$ is $⟨S, I, T, P, L⟩$, where $S$ is a set of states, $I ⊆ S$ is the set of initial states, $T ⊆ S × S$ is a total binary relation over $S$, $P$ is a set of atomic propositions and $L$ is a labeling function whose type is $S → 2^P$. Each element $(s, s′) ∈ T$ is called a state transition from $s$ to $s′$ and $T$ may be called the state transitions (with respect to $K$). For a state $s ∈ S$, $L(s)$ is the set of atomic propositions that hold in $s$. A path $π$ is an infinite sequence $s_0, s_1, s_2, …$ of states such that $s_i ∈ S$ and $(s_i, s_{i+1}) ∈ T$ for each $i$. Let $π^i$ be $s_i, s_{i+1}, …$ and $π(i)$ be $s_i$. Let $P$ be the set of all paths. $π$ is called a computation if $π(0) ∈ I$. Let $C$ be the set of all computations.

The syntax of a formula $ϕ$ in LTL for $K$ is $ϕ ::= ⊤ | p | ¬ϕ | ϕ ∧ ϕ | ϕ U ϕ$, where $p ∈ P$. Let $F$ be the set of all formulas in LTL for $K$. An arbitrary path $π ∈ P$ of $K$ and an arbitrary LTL formula $ϕ ∈ F$ of $K$, $K, π ⊨ ϕ$ is inductively defined as $K, π ⊨ ϕ$ if $p ∈ L(π(0))$, $K, π ⊨ ¬ϕ$ if $K, π ⊨ ϕ_1$, $K, π ⊨ ϕ_1 ∧ ϕ_2$ if $K, π ⊨ ϕ_1$ and $K, π ⊨ ϕ_2$, $K, π ⊨ ϕ_1 U ϕ_2$ if there exists a natural number $i$ such that $K, π^i ⊨ ϕ_2$ and for all natural numbers $j < i$, $K, π^j ⊨ ϕ_1$. The temporal connectives $∧$ and $U$ are called the next connective and the until connective, respectively. The other logical and temporal connectives are defined as usual as follows: $⇒ ⊢ ¬⇒, ϕ_1 ∧ ϕ_2 ⊢ ¬(ϕ_1 ∧ ¬ϕ_2), ϕ_1 ⇒ ϕ_2 ⊢ ¬ϕ_1 ∨ ϕ_2, Oϕ ⊢ T ϕ$, and $□ϕ ⊢ ¬(Δ¬ϕ)$. The temporal connectives $Δ$ and $□$ are called the eventually connective and the always connective, respectively.

In this paper, to express a state of $S$, we use an associative-commutative collection of name-value pairs. Associative-commutative collections are called soups, and name-value pairs are called observable components. That is, a state is expressed as a soup of observable components. The juxtaposition operator is used as the constructor of soups. An analyzer and an LTL model checker. A rewrite rule starts is expressed as a soup of observable components. The also equipped with model checking facilities (a reachability analyzer and an LTL model checker). A rewrite rule starts on rewriting logic, to specify them as rewrite rules. Maude [5], a programming/specification language based

where $lb$ is a label and $c_i$ is part of the condition, which may be an equation $lc_i = rc_i$. The negation of $lc_i = rc_i$ could be written as $(lc_i /= rc_i) = true$, where $true$ could be omitted. If the condition ... /\ $c_i$ /\ ... holds under some substitution $σ$, $σ(l)$ can be replaced with $σ(r)$.

Let init be the only initial state of $K$ and $ϕ$ be an LTL formula. Then, the Maude LTL model checker checks that $K$ satisfies $ϕ$ by the following command:

```
red modelCheck(init, ϕ) .
```

where red is an abbreviation of reduce. Executing this command, Maude will return either true if $ϕ$ is satisfied, or a counterexample when $ϕ$ is not satisfied.

III. The Recoverable Wait-Free MCS Protocol

The pseudo-code of RWfMCS protocol for each process $i$ can be written as follows:

```
re : if state[i] = LEAVE then goto ex1;
else if state[i] = TRY and pred[i] = mine[i]
then goto ex1;
else if state[i] = FREE then 
  state[i] := INIT; mine[i] := null;
end
en1 : if state[i] = INIT 
  if mine[i] = null then mine[i] := newNode();
end
en2 : if nextmine[i] = null then lockmine[i] := true;
end
en3 : nextmine[i] := mine[i]; state[i] := TRY;
end
en4 : if state[i] = TRY then 
  if pred[i] = mine[i] 
    temp := FAS(tail, mine[i]);
end
en5 : if pred[i] = mine[i] 
  pred[i] := temp; | crash and goto re; 
end
en6 : if pred[i] ≠ null 
  if CAS(nextpred[i], null, mine[i])
end
en7 : repeat while lockmine[i];
end
en8 : if pred[i] ≠ null 
  cs := state[i] := InCS; 
end
ex1 : state[i] := LEAVE; CAS(tail, mine[i], null);
end
ex2 : if not CAS(nextmine[i], null, mine[i])
end
ex3 : locknextmine[i] := false;
ex4 : state[i] := FREE; goto re;
end
```

RWfMCS maintains a queue of processes based on a linked list. That is why we call the queue is the virtual queue. Each element of the linked list is a node that contains the following two fields:

- next: stores the address of its successor (or next) node in the virtual queue if any and null otherwise.
- lock: stores a Boolean value. A process needs to spin to wait for its turn while trying to enter the critical section if its lock value is true.

next$i$ and lock$i$ can be regarded as the local variables of process $i$. RWfMCS uses the following global variables (i.e., shared among all processes):

- pred: an array where each element pred[i] contains an address referring to the predecessor node of process $i$ in the virtual queue if any and null otherwise.
Whenever a process crashes, it loses the information about its

The body of the loop at the label

If

tail

to wait until its

for the lock. In that case, the process releasing the lock needs

is created by the successor process, then the exit segment

code of the original MCS algorithm is not wait-free. To

overcome that problem, RWfMCS is augmented with some
modifications at labels ex2 and en9. At label ex2, a process
leaving the critical section sets the next field of it to itself if
the link from it to its successor process in the virtual queue
has not been created yet in order to inform the next process
in the queue that the lock is now free. While at label en9,
by performing the CAS instruction, the next process in the
queue checks the value of the next field of its predecessor
is null or not. If the value is null, indicating that the lock is
now free, then the process can enter the critical section. On
the other hand, it creates the link between its predecessor
and itself, and spins to wait until its lock becomes false.

IV. FORMAL SPECIFICATION THE PROTOCOL

In this paper, a state is expressed as a soup of observable
components. To formalize RWfMCS as a Kripke structure
\( K_{\text{MCS}} \), we use the following observable components:

- \((\text{tail} : p)\) - it says that tail is p,
- \((\text{pc}[p] : l)\) - it says that process p is located at label l,
- \((\text{next}[p] : q)\) - it says that nextp refers to q,
- \((\text{lock}[p] : b)\) - it says that lockp is b,
- \((\text{pred}[p] : q)\) - it says that predp refers to q,
- \((\text{state}[p] : s)\) - it says that statep is s,
- \((\text{mine}[p] : q)\) - it says that minep refers to q,
- \((\text{temp}[p] : q)\) - it says that tempp refers to q,

where p and q are two process IDs, l receives one of the
sixteenth label values, b is a Boolean value, s receives one of
five values of the state. Although in the pseudo-code,
temp is used as a temporary variable, in the specification,
we explicitly use different temp for each process to avoid
undesirable behavior caused by jointly reading/writing temp.

Each state in \( S_{\text{MCS}} \) is expressed as \{obs\}, where obs is a
soup of those observable components. If two processes p1
and p2 participate in RWfMCS, one initial state of \( I_{\text{MCS}} \)

namely \( \text{init} \) is defined as follows:

\[
\{(\text{tail} : \text{null}) \ (\text{pc}[p1] : \text{re}) \ (\text{pc}[p2] : \text{re}) \\
(\text{next}[p1] : \text{null}) \ (\text{next}[p2] : \text{null}) \\
(\text{lock}[p1] : \text{false}) \ (\text{lock}[p2] : \text{false}) \\
(\text{pred}[p1] : \text{null}) \ (\text{pred}[p2] : \text{null}) \\
(\text{mine}[p1] : \text{null}) \ (\text{mine}[p2] : \text{null}) \\
(\text{temp}[p1] : \text{null}) \ (\text{temp}[p2] : \text{null}) \\
(\text{state}[p1] : \text{FREE}) \ (\text{state}[p2] : \text{FREE}) \} .
\]

There are seventeen transitions for each process p:

- \( \text{rcv} : p \) performs the recover action (if crashed before)
and moves to either en1 or ex1 from re,
- \( \text{chsta} : p \) checks the if condition at en1 and moves to
either en2 or en4 from en1,
- \( \text{initmine} : p \) moves to en3 from en2,
- \( \text{init} : p \) moves to en4 from en3,
- \( \text{chsta2} : p \) checks the if condition at en4 and moves to
either en5 or ex1 from en4,
- \( \text{chprd} : p \) checks the if condition at en5 and moves to
either en6 or en8 from en5,
The rewrite rule says that when a process $P$ asks to exit specifying $\text{RWfMCS}$ is how to specify dynamic memory allocation. The rewrite rule is simply defined as follows:

$$\text{rl} [\text{initMine}] : \{(\text{mine}[P] : Q) (\text{nodes} : (Q1 \text{LP})) (\text{nodes} : (q1 q2 q3 q4 q5 q6)) \} \Rightarrow (\text{pc}[P] : (\text{en2}) (\text{mine}[P] : \text{null}) (\text{nodes} : \text{empty}) \text{OCs}) \text{.}$$

Every time a process requests for allocating memory to construct a new empty node (i.e., calls to the function $\text{newNode()}$ at label $\text{en2}$), a top node of the list is extracted and used. When a process makes a request for a new node but the list of nodes now is empty, we let the process move to the terminal state in which the process spins there forever. We add one more observable component $(\text{nodes} : \text{lp})$, where $\text{lp}$ is a list of process IDs, to represents the list of “empty nodes” used for dynamic allocation. Consequently, we add the following observable component to $\text{init}$:

$$(\text{nodes} : (q1 q2 q3 q4 q5 q6)) \text{.}$$

where $q_k$ is a process ID for each $k = 1, \ldots, 6$. Here we provide six “empty nodes” for dynamic allocation. The rewrite rule $\text{initMine}$ now is defined as follows:

$$\text{rl} [\text{initMine}] : \{(\text{mine}[P] : Q) (\text{nodes} : (Q1 \text{LP})) (\text{pc}[P] : \text{en2}) \text{OCs} \Rightarrow \{(\text{pc}[P] : \text{en3}) (\text{mine}[P] : (\text{if } Q == \text{null then } Q1 \text{ else } Q \text{ fi}) \}) \text{nodes} : (\text{if } Q == \text{null then } \text{LP else } (Q1 \text{ LP}) \text{ fi}) \text{OCs} \} \text{.}$$

Let OCs be a Maude variable of observable component soups, $F$, $Q$, $Q1$ be Maude variables of process IDs, and $S$ be Maude variable receives one of five values of the state. The rewrite rule $\text{exit}$ is simply defined as follows:

$$\text{rl} [\text{exit}] : \{(\text{pc}[P] : \text{cs}) (\text{state}[P] : S) \text{OCs} \Rightarrow \{(\text{pc}[P] : \text{ex1}) (\text{state}[P] : \text{InCS}) \text{OCs} \} \text{.}$$

The rewrite rule says that when a process $P$ is located at $\text{cs}$, $P$ moves to $\text{ex1}$; and $\text{state}[P]$ changes to $\text{InCS}$; other observable components do not change.

The rewrite rule $\text{rcv}$ is defined as follows:

$$\text{rl} [\text{rcv}] : \{(\text{pc}[P] : \text{re}) (\text{state}[P] : S) (\text{pred}[P] : Q) (\text{mine}[P] : Q1) \text{OCs} \Rightarrow \{(\text{pc}[P] : (\text{if } S == \text{LEAVE} \text{ then } \text{ex1} \text{ else } (\text{if } S == \text{TRY} \text{ and } Q == \text{Q1} \text{ then } \text{ex1} \text{ else } \text{en1} \text{ fi}) \text{ fi}) (\text{state}[P] : (\text{if } S == \text{FREE} \text{ then } \text{INIT} \text{ else } S \text{ fi}) (\text{mine}[P] : (\text{if } S == \text{FREE} \text{ then } \text{null} \text{ else } Q1 \text{ fi}) (\text{pred}[P] : Q) \text{OCs}) \} \text{.}$$

The rewrite rule says that when a process $P$ located at $\text{re}$, if its $\text{state}$ is $\text{LEAVE}$ or its $\text{state}$ is $\text{TRY}$ and its $\text{pred}$ equals to its $\text{mine}$, $P$ then moves to $\text{ex1}$, otherwise, $P$ moves to $\text{en1}$; if its $\text{state}$ is $\text{FREE}$, its $\text{state}$ changes to $\text{INIT}$ and $\text{mine}[P]$ is reset to null, otherwise, nothing changes; other observable components do not change.

One challenge we need to deal with during formally specifying $\text{RWfMCS}$ is how to specify dynamic memory allocation. In the algorithm, when the function $\text{newNode()}$ at label $\text{en2}$ is invoked, a new memory location is allocated from which an empty node is constructed and assigns to $\text{mine}[i]$. After the process $i$ successfully enters the critical section, releases the lock, and goes back to the recover section, $\text{mine}[i]$ is reset to null (at label $\text{re}$). This assignment simply points $\text{mine}[i]$ to a null pointer, but the memory that contains the old node $\text{mine}[i]$ is still alive without any effect. Roughly speaking, the values returned by the function $\text{newNode()}$ are different from time to time every time process $i$ requests for allocating memory to construct a new empty node. Furthermore, resetting the value of $\text{mine}[i]$ to null does not affect the old value of $\text{mine}[i]$. It is, however, not simple to make the formal specification satisfying those behaviors.

To solve the problem of formally specifying dynamic memory allocation mentioned above, our solution is to provide a fixed list of “empty nodes” from the beginning. Every time a process requests for allocating memory to construct a new empty node (i.e., calls to the function $\text{newNode()}$ at label $\text{en2}$), a top node of the list is extracted and used. When a process makes a request for a new node but the list of nodes now is empty, we let the process move to the terminal state in which the process spins there forever.

We add one more observable component $(\text{nodes} : \text{lp})$, where $\text{lp}$ is a list of process IDs, to represents the list of “empty nodes” used for dynamic allocation. Consequently, we add the following observable component to $\text{init}$:

$$(\text{nodes} : (q1 q2 q3 q4 q5 q6)) \text{.}$$

where $q_k$ is a process ID for each $k = 1, \ldots, 6$. Here we provide six “empty nodes” for dynamic allocation. The rewrite rule $\text{initMine}$ now is defined as follows:

$$\text{rl} [\text{initMine}] : \{(\text{mine}[P] : Q) (\text{nodes} : (Q1 \text{LP})) (\text{pc}[P] : \text{en2}) \text{OCs} \Rightarrow \{(\text{pc}[P] : \text{en3}) (\text{mine}[P] : (\text{if } Q == \text{null then } Q1 \text{ else } Q \text{ fi}) \}) \text{nodes} : (\text{if } Q == \text{null then } \text{LP else } (Q1 \text{ LP}) \text{ fi})) \text{OCs} \} \text{.}$$

where $\text{LP}$ is a Maude variable whose value is a list of process IDs (possibly empty). The rewrite rule says that when a process $P$ is located at $\text{en2}$ and $\text{nodes}$ is not empty (i.e., consists of $Q1$ and $\text{LP}$), $P$ moves to $\text{en3}$; if $\text{mine}[P]$ is null then two assignments are performed: assigning the top element of nodes (i.e., $Q1$) to $\text{mine}[P]$, and updating nodes by removing its top element.

We need to add a new rewrite rule to represent the transition when a process requests for allocating a new node but nodes now is empty. The rewrite rule is defined as follows:

$$\text{rl} [\text{terminate}] : \{(\text{pc}[P] : \text{en2}) (\text{mine}[P] : \text{null}) (\text{nodes} : \text{empty}) \text{OCs} \Rightarrow \{(\text{pc}[P] : \text{terminal}) \text{mine}[P] : \text{null}) (\text{nodes} : \text{empty}) \text{OCs}) \text{.}$$

where $\text{terminal}$ is a new process location in addition to the sixteenth existing locations. When a process moves to $\text{terminal}$, it will stay there forever by the stutter rewrite rule that is defined as follows:

$$\text{rl} [\text{stutter}] : \{(\text{pc}[P] : \text{terminal}) \text{OCs} \Rightarrow \{(\text{pc}[P] : \text{terminal}) \text{OCs}) \text{.}$$

The remaining transitions can be defined likewise.

V. Model checking

A. Model checking without any restriction to crashes

To model check that $K_{MCS}$ satisfies some desired properties, we define $P_{MCS}$ and $L_{MCS}$. $P_{MCS}$ contains an atomic proposition namely $\text{inCs}$ which takes a process IDs as its argument. $L_{MCS}$ is initially specified as follows:

$$\text{eq } \{(\text{pc}[P] : \text{cs}) \text{OCs} \} \Rightarrow \text{inCs}(P) = \text{true} \text{.}$$

$$\text{eq } \{\text{OCs} \} \Rightarrow \text{PROP} = \text{false} \text{[owise]} \text{.}$$
where \textit{owise} is the abbreviation of otherwise, indicating that this equation will only be applied if all of the previous equations above it can not be applied. The equations say that inCs\(_{(P)}\) holds in a state \(s\) iff \(s\) contains \((pc[\text{P}] : cs)\).

We then specify the mutual exclusion property as the following LTL formula:

\[
eq\text{mutex} = (\Box (\neg (\text{inCs}(p1) \lor \neg \text{inCs}(p2)))).
\]

where \(\Box\) is \(\square\), \(\sim\) is \(\neg\), and \(\lor\) is \(\land\). The equation (or formula) says that it is always the case such that \(p1\) and \(p2\) are not located at \(cs\) at the same time. We use Maude model checker to check that RWfMCS satisfies the mutual exclusion property or not by using the following Maude command:

\[
\text{red \ modelCheck(init,\text{mutex})}.
\]

Unfortunately, a counterexample was found, which is visualized as in Fig. 1. Note that, the Figure does not show all observable components, but only depicts \(pc[p1]\) and \(pc[p2]\), and uninteresting transitions are omitted (e.g., \(\text{chsta, initmine}\)). When the virtual queue consists of two processes \(p1\) and \(p2\) such that \(p1\) is located at \(cs\), \(p2\) is located at \(en7\), and \(tail\) is \(p2\), process \(p2\) crashes and goes back to \(re\). Because \(\text{state[p2]}\) now is TRY, it then jumps to \(ex1\), completes the exit segment in which it sets \(tail\) to null by the CAS instruction. \(p2\) then tries to enter the critical section one more time. Since \(tail\) is null now, if only transitions of \(p2\) are executed until it reaches \(en8\) (without crash again), \textit{pred[p2]} will be null. That time, \(p2\) gets permission to directly enter \(cs\), leading to the mutual exclusion property is not satisfied since there are two different processes \(p1\) and \(p2\) located at the critical section.

The first experiment says that if there is not any restriction to the occurrence of crashes, RWfMCS does not enjoy the mutual exclusion property. In the upcoming subsection, we report model checking under some assumptions of the occurrence of crashes.

\subsection{B. Model checking under crash assumptions}

\subsubsection{First assumption: crashes never happen at all}

We add the following observable component to keep track of the occurrence of crashes: \((\text{crash}: b)\), where \(b\) is a Boolean value. Then, the rewrite rule \(\text{crash}\) is modified to become as follows:

\[
r1 \ [\text{crash}] : \{(pc[\text{P}] : en7) \ (next[\text{P}] : Q) \ (lock[\text{P}] : B) \ (\text{crash} : B1) \ OCs} \Rightarrow \{(pc[\text{P}] : re) \ (next[\text{P}] : null) \ (lock[\text{P}] : false) \ (\text{crash} : true) \ OCs}.
\]

where \(B\) and \(B1\) are Maude Boolean variables. The rewrite rule says that when a process \(\text{P}\) is located at \(en7\), it may crash then go back to \(re\), \text{crash} is set to \(true\), and \(\text{P}\) loses all information about its \text{next} and \text{lock}. \text{crash} does not change in other transitions, and initially, it is set to \(false\).

One more atomic proposition namely \text{crashed} is added into \(PMCS, LMCS\) is modified by adding the following equation before the existing \textit{owise} statement at the end.

\[
\text{eq} \{(\text{crash} : true) \ OCs\} \models \text{crashed} = \text{true}.
\]

The equation says that \text{crashed} holds in a state \(s\) iff \(s\) contains \((\text{crash} : \text{true})\). Since \text{crash} never can be changed back to \text{false} from \text{true}, we can say that when there is not any crash so far, \text{crashed} will not hold, otherwise, it will. We model check the mutual exclusion property under the assumption that crashes never happen at all by using the following command:

\[
\text{red \ modelCheck(init,([\sim \text{crashed}) \ -> \ \text{mutex})}.
\]

No counterexamples were found. It took about 13 seconds for Maude to complete the model checking. Consequently, we can conclude that the protocol enjoys the mutual exclusion property if crashes never happen.
Second assumption: “crashes have not occurred recently”

The assumption that crashes never happen seems so strong, we should check it under a weaker assumption. In this section, we model check that RWfMCS enjoys the mutual exclusion property under the assumption: “crashes have not occurred recently”. The key idea of this assumption is that after a crash, crash can be set back to false from true if all processes have state FREE. It means that after a crash, all requests for entering the critical section before are satisfied, implying the behavior of the protocol backs to normal as without failures. To check the condition whether all processes have state FREE, we need to introduce one more observable component namely (noPsFree: n), where n is a natural number. noPsFree maintains the number of processes that have state FREE. Initially, noPsFree is set to the number of processes participating in the protocol. When a process P is located at re and state[P] is FREE, the transition rcv will change its state to INIT, then noPsFree needs to be decreased by one. Thus, the rewrite rule rcv is modified to become as follows:

\[
\text{rcv: } \{ (pc[P]: re) (state[P]: S) \} \Rightarrow \{ (pc[P]: (if S \equiv LEAVE then exl else (if S \equiv TRY and Q \equiv Q1 then exl else enl fi)) (state[P]: (if S \equiv FREE then INIT else S fi)) (mine[P]: (if S \equiv FREE then null else Q1 fi)) (noPsFree: (if S \equiv FREE then dec(N) else N) fi)) (pred[P]: Q) OCs \} .
\]

where N is a Maude variable of natural numbers, and dec(N) is a function that decreases N by one.

noPsFree needs to be incremented when a process updates its state to FREE. In the algorithm, transition go2rcv is the only one that can change the state of a process to FREE. Therefore, the rewrite rule go2rcv is revised as follows:

\[
\text{go2rcv: } \{ (pc[P]: ex4) (state[P]: S) \} \Rightarrow \{ (pc[P]: re) (noPsFree: N) (crash: B) OCs \} \}
\]

In addition to updating noPsFree, the rewrite rule also sets the value of crash back to true if the value of noPsFree after increasing is equal to the total number of processes participating in the protocol (i.e., 2 in our case).

One more atomic proposition namely ncr (not recently crash) is added into \( L_{MCS} \) to express the assumption that crashes have not occurred recently. \( L_{MCS} \) is modified by adding the following equation before the existing owise statement at the end.

\[
\text{eq \{ crash: false \} OCs \} \Rightarrow \{ nrc \equiv (\neg (\text{oise}(p1) \lor \text{oise}(p2))) \} .
\]

No counterexamples were found. It took about 5 minutes for Maude to complete the model checking. Consequently, we can conclude that the protocol enjoys the mutual exclusion property if crashes have not occurred recently.

VI. CONCLUSION

We have presented model checking the recoverable wait-free version of MCS protocol. The recoverable property indicating that the protocol still works under the assumption that crashes may occur during the execution of processes. The wait-free property says that exit segment code of the new version is wait-free (i.e., always completes in a finite step regardless of the behavior of other processes). The first experiment shows a counterexample in which the protocol does not satisfy the mutual exclusion property. Analyzing the counterexample, we could understand the scenario leading to two different processes located at the critical section at the same time if crashes are allowed to happen without any restriction. The second experiment says that the protocol enjoys the mutual exclusion property if crashes never happen. The last experiment says that the protocol enjoys the mutual exclusion property if crashes have not occurred recently. We have also described the challenge of how to formally specify dynamic allocation and presented our solution to solve that problem during formal specifying the protocol.

One piece of our future work is to model check the protocol satisfies the lock-out freedom property in particular or other liveness properties in general. Model checking such properties usually requires some fairness assumptions. However, the formula to model check often becomes very complicated if some fairness assumptions are included, leading to the model checker could not terminate after a reasonable amount of time. One possible way to make it feasible is by using the technique presented in [6].

REFERENCES

Fine-Grained Neural Network Abstraction for Efficient Formal Verification

Zhaosen Wen¹, Weikai Miao¹², Min Zhang¹²

¹ Shanghai Key Laboratory for Trustworthy Computing, East China Normal University
² Shanghai Institute of Intelligent Science and Technology, Tongji University
51184501057@stu.ecnu.edu.cn, {wkmiao,zhangmin}@sei.ecnu.edu.cn

Abstract
The advance of deep learning makes it possible to empower safety-critical systems with intelligent capabilities. However, its intelligent component, i.e., deep neural network, is difficult to formally verify due to the large scale and intrinsic complexity of the verification problem. Abstraction has been proved to be an effective way of improving the scalability. A challenging problem in abstraction is that it is difficult to achieve a balance between the size reduced and output overestimation caused by abstraction. In this work, we propose an effective fine-grained approach to abstract neural networks. Our approach is fine-grained in that we identify four cases that should be abstracted independently under a certain neuron prioritization strategy. This allows us to merge more neurons in networks and meanwhile maintain a relatively low output overestimation. Experimental results show that our approach outperforms other existing abstraction approaches by significantly reducing the scale of target deep neural networks with small overestimation.

1 Introduction
In recent years, Deep Neural Networks (DNNs) have been achieving remarkable performance in many complex tasks and are increasingly deployed in safety-critical systems, such as autonomous vehicle [2], face recognition [3], airborne collision avoidance system [11]. However, it is well known that DNNs are vulnerable to slight perturbations, i.e., adding imperceptible perturbations to inputs may cause DNN to make mistakes [17, 18, 8, 5]. Statistical results show that the accident frequency of autonomous vehicles is much higher than that of conventional vehicles [7]. As safety-critical systems require strict safety and reliability guarantees, it raises a new problem of certifying the trustworthiness of the intelligent components, i.e., DNNs.

Formal methods have been proved their effectiveness in certifying DNNs. For rigorousness, formal methods guarantee a DNN satisfies a property if the property is proven to be true, and otherwise counterexamples are computed as witnesses to the violation. In the context of neural network verification, a counterexample is called an adversarial example that causes misclassification to DNN. The literature on formal verification of neural network is booming in the past several years. Details can be referred to the survey [10].

Most of the existing neural network verification approaches suffer from bad scalability issue due to the intrinsic complexity of neural networks. Katz et al. [12] showed that the verification problem of even simple fully connected feedforward neural networks taking ReLU activation function is NP-complete. Abstraction is one of the effective approaches to scale up verification algorithms. The basic idea of abstraction is to tune concrete constraints into abstract ones which can be solved more efficiently [9, 15, 8, 1, 6, 13]. Abstraction must preserve soundness, i.e., a property proved in the abstract system implies the concrete system satisfies that property.

One promising abstraction technique for neural network verification is to construct Interval Neural Networks (INNs) [13] to abstract ordinary deep neural networks. Intuitively, an INN takes intervals as inputs, unlike DNN whose inputs are concrete values. An INN can be constructed by merging neurons in the same hidden layer [13]. The decrease of neurons makes it faster to verify an INN than to verify its corresponding DNN. There are two criteria for evaluating an abstraction approach, i.e., the number of neurons that are merged, and the overestimation of output interval. An approach that can merge more neurons with lower overestimation is more preferred than the one that merges fewer neurons with larger overestimation.

In this paper, we propose a novel fine-grained abstraction approach to abstract feedforward neural networks that take ReLU as activation function into INNs for the purpose of improving the efficiency of their formal verification. In our approach, we classify the merging of neurons into four cases according to the signs of weights, and propose the corresponding merging rules for each case. We also devise a strategy for determining the priority of neurons to be merged. Before the abstraction of neural network, we compute an indicator for each pair of neurons and priori-
tize neurons according to the indicator. We prove that our abstraction approach is sound. We implement the approach into a tool called NNZipper, and evaluate it on the benchmark of neural networks trained on MNIST [16] and ACAS Xu [11]. Experimental results show that our approach can significantly reduce the scale of a neural network with low overestimation induced, compared with the pioneering abstraction approach in [13].

2 Preliminaries

2.1 Deep Neural Network (DNN)

A deep neural network is a model consisting of an input layer, several hidden layers and an output layer. Except for input layer, each layer contains some neurons, which are connected to the neurons in the preceding layer. Each edge has a weight. The neurons in input layer receive input data, and the neurons in the next layer get their values by computing a dot product of the values of preceding layer and edge weights, with the addition of a bias, and then operated by an activation function such as ReLU. After layer-by-layer calculation, the output layer gives the result of DNN.

Definition 1 (DNN). An n-layer DNN is a triple \((S_i)_{1 \leq i \leq n}, (W_i)_{1 \leq i \leq n}, (B_i)_{1 \leq i \leq n}\), where

- \(S_i\) is the set of neurons in the i-th layer. \(S_0\) denotes the input layer, \(S_n\) denotes the output layer.
- \(W_i\) denotes the weight matrix between the i-1-th layer and i-th layer.
- \(B_i\) denotes biases vector of the i-th layer.

2.2 Interval Neural Network (INN)

In an interval neural network (INN), the edge weights and biases are not in value form but interval form.

Definition 2 (INN [13]). An n-layer INN is a triple \((S_i)_{0 \leq i \leq n}, (W^l_i, W^u_i)_{1 \leq i \leq n}, (B^l_i, B^u_i)_{1 \leq i \leq n}\), where

- \(S_i\) is the set of neurons in the i-th layer. \(S_0\) denotes the input layer, \(S_n\) denotes the output layer.
- \(W^l_i, W^u_i\) denote the lower weight matrix and upper weight matrix respectively between the i-1-th layer and i-th layer, which satisfy \(W^l_i \leq W^u_i\).
- \(B^l_i, B^u_i\) denote the lower biases vector and upper biases vector respectively of the i-th layer.

The input of INN is a set of intervals, as well as the output. The output is computed by solving several maximization and minimization problems built on the lower and upper weights and biases. It is proved that a DNN can be abstracted to an INN with smaller size [13]. For an identical input region, the output of the INN is an over-approximation of the output range of the original DNN.

3 Fine-Grained Abstraction

3.1 Abstracting DNN into INN

The complexity of the output range computation of a DNN is strongly related to its size; i.e., the number of all the neurons in the DNN. Our abstraction method aims to decrease the size of the network, and get an over-approximation of the network’s output range by computing the abstract network’s output range. To accomplish this, several pairs of neurons in the same hidden layer are merged into a single neuron with their weights and biases merged into intervals. The state-of-the-art approach takes the convex hull of the original weights as the weight interval for the neuron abstracted from original neurons [13], which is valid and fast but induces considerable imprecision in output range computation.

Figure 1 gives a simple example of abstracting a DNN into an INN by merging neurons and weights. When merging outgoing weights, the convex hull of original weights needs to be multiplied by 2 (equal to the number of neurons merged) to guarantee validity.

3.2 Neural Network Abstraction

Neural network abstraction is a technique for compressing a neural network into a smaller one. Despite the loss of some precision, a smaller network is usually preferred to deploy on edge devices and to formally verify. To abstract a DNN into a smaller INN, several neurons in the same hidden layer are merged with their weights and biases merged into intervals. The state-of-the-art approach takes the convex hull of the original weights as the weight interval for the neuron abstracted from original neurons [13], which is valid and fast but induces considerable imprecision in output range computation.

Figure 1 gives a simple example of abstracting a DNN into an INN by merging neurons and weights. When merging outgoing weights, the convex hull of original weights needs to be multiplied by 2 (equal to the number of neurons merged) to guarantee validity.

3.3 Abstracting DNN into INN

The complexity of the output range computation of a DNN is strongly related to its size; i.e., the number of all the neurons in the DNN. Our abstraction method aims to decrease the size of the network, and get an over-approximation of the network’s output range by computing the abstract network’s output range. To accomplish this, several pairs of neurons in the same hidden layer are merged into a single neuron with their weights and biases also merged. The new neuron’s weights and bias are not value but intervals, which are calculated based on the weights and biases of original neurons. Note that DNN can be regarded as a special kind of INN whose weights and biases are degenerate intervals, i.e., the lower bound and upper bound of the interval are the same. Hence we will describe the details of the abstraction based on the semantics of INN.

Given an INN \((n, \{S_i\}_{0 \leq i \leq n}, \{W^l_i, W^u_i\}_{1 \leq i \leq n}, \{B^l_i, B^u_i\}_{1 \leq i \leq n}\)

let \(s_{i,p}\) denote the p-th neuron in layer i, and \(w^l_{i,p}, w^u_{i,p}\) denote the lower weight vector and upper weight vector of neuron \(s_{i,p}\), and \(w^l_{i,pq}, w^u_{i,pq}\) denote the lower weight and upper
weight between the $q$-th neuron in layer $i - 1$ and the $p$-th neuron in layer $i$. We use $v_{i,p}$ to denote the valuation interval of the $p$-th neuron in layer $i$, and $b_{i,p}^l, b_{i,p}^u$ to denote the lower bias and upper bias of the $p$-th neuron in layer $i$. The problem of abstracting a DNN by an INN is merging two neurons $s_{i,\alpha}$ and $s_{i,\beta}$ into a new neuron $s_{i,\gamma}$ ($1 \leq i \leq n - 1$).

Our first step is to merge the biases of $s_{i,\alpha}$ and $s_{i,\beta}$ and the edges between layer $i - 1$ and layer $i$. The requirement of this step is guaranteeing the valuation interval of the new neuron containing the valuation interval of the original neurons, i.e., $v_{i,\alpha} \subseteq v_{i,\gamma}, v_{i,\beta} \subseteq v_{i,\gamma}$. To reach this goal, the new neuron’s bias interval is obtained by taking the convex hull of the two original neurons’ bias intervals. Specifically, the smaller of the original lower bounds will be the new lower bound, and the greater of the original upper bounds will be the new upper bound. The merging of the edge weights is similar. For each new edge between layer $i - 1$ and layer $i$, the new weight interval of the edge is obtained by taking the convex hull of the corresponding original weight intervals. Formally, we have the following equations:

**Bias**: $b_{i,\gamma}^l = \min(b_{i,\alpha}^l, b_{i,\beta}^l), b_{i,\gamma}^u = \max(b_{i,\alpha}^u, b_{i,\beta}^u)$;

**Weights**: $\forall_{S_{i-1, p} \in S_{i-1}, w_{i,\gamma}^{l,p} = \min(w_{i,\alpha}^{l,p}, w_{i,\beta}^{l,p}), w_{i,\gamma}^{u,p} = \max(w_{i,\alpha}^{u,p}, w_{i,\beta}^{u,p})$.

Figure 2 shows a simple example of the first step. After the merging, the new weight interval $[-1, 2]$ is the convex hull of $[-1, -1]$ and $[1, 2]$, and the other new weight interval $[1, 3]$ is similar. The new bias interval $[-1, 1]$ is also the convex hull of the original bias intervals $[1, 3]$ and $[-1, 1]$.

Our second step is to merge the edges between layer $i$ and layer $i + 1$. The requirement of this step is guaranteeing the new valuation interval of each neuron in layer $i + 1$ containing its original valuation interval, i.e., $\forall_{S_{i+1, q} \in S_{i+1}, v_{i+1, q}^{l,p} \supseteq v_{i+1, q}}$. According to the sign of the weights of original neurons, the merging in this step will follow different rules. Without loss of generality, we assume that the lower bound of the weight interval for $s_{i,\beta}$ is not greater than that of $s_{i,\alpha}$, i.e., $w_{i+1, q, \alpha}^l \leq w_{i+1, q, \beta}^l$. Then the merging can be classified into four cases:

**Case 1** We first consider the case when $w_{i+1, q, \alpha}^l, w_{i+1, q, \beta}^l$ have the same sign, and $w_{i+1, q, \alpha}^u, w_{i+1, q, \beta}^u$ have the same sign. The lower bound of the new weight interval is obtained by taking the sum of the two original lower bounds. Formally, we have:

$$w_{i+1, q, \gamma}^l = w_{i+1, q, \alpha}^l + w_{i+1, q, \beta}^l$$

**Case 2** In the second case, we consider that $w_{i+1, q, \alpha}^l < 0$, $w_{i+1, q, \beta}^l < 0$, $w_{i+1, q, \beta}^u \geq 0$, and $w_{i+1, q, \alpha}^u \geq 0$. In this case, the lower bound of the new weight interval is equal to the lower bound of the original weight interval for $s_{i,\alpha}$. The upper bound of the new weight interval is equal to the original upper bound for $s_{i,\beta}$. Formally, they are defined as follows:

$$w_{i+1, q, \gamma}^u = w_{i+1, q, \alpha}^u$$

**Case 3** The third case considers $w_{i+1, q, \alpha}^l < 0$, $w_{i+1, q, \alpha}^u \geq 0$, $w_{i+1, q, \beta}^l \geq 0$, and $w_{i+1, q, \beta}^u \geq 0$. In this case, the lower bound of the weight interval after merging is equal to the lower bound of the original weight interval for $s_{i,\alpha}$. The upper bound of the new weight interval is the sum of the two original upper bounds. Formally, we have:

$$w_{i+1, q, \gamma}^l = w_{i+1, q, \alpha}^l$$

**Case 4** The last case is that $w_{i+1, q, \alpha}^l < 0$, $w_{i+1, q, \alpha}^u < 0$, $w_{i+1, q, \beta}^l < 0$, and $w_{i+1, q, \beta}^u \geq 0$. In this case, the lower bound of the new weight interval takes the sum of the two original lower bounds. And the upper bound of the new weight in-
interval is equal to the original upper bound for \( s_{i,b} \). Formally, they are defined by the following equations:

\[
\begin{align*}
    w_{i+1,q}^{f} &= w_{i+1,q}^{f} + w_{i+1,q}^{f} \\
    w_{i+1,q}^{u} &= w_{i+1,q}^{u} + w_{i+1,q}^{u}
\end{align*}
\]

Figure 3d shows an example of the last case. The new upper bound is 1, which is derived from the difference of the weights and biases.

**Example 1.** Let us consider the neural network in Figure 4a. We want to merge \( s_{2,1} \) and \( s_{2,2} \), and then \( s_{1,1} \) and \( s_{1,2} \). For both mergings, the new weight intervals of the preceding edges are obtained by taking the convex hull of the corresponding original weights. We focus on the merging of succeeding edges. In the merging of \( s_{2,1} \) and \( s_{2,2} \), because \( 1 > 0, -1 < 0 \), the weight interval \([-1,1]\) between \( s'_{2,1} \) and \( s_{3,1} \) is obtained by applying the rules of case 2. And the weight \( -5 \) between \( s'_{2,1} \) and \( s_{3,2} \) is obtained by applying the rules of case 1 because \(-2 \) and \(-3 \) are both negative. Then in the merging of \( s_{1,1} \) and \( s_{1,2} \), the weight interval \([-1,4]\) is obtained by applying the rules of case 3, and the weight 4 by applying the rules of case 1.

Our method can abstract a DNN into an INN with arbitrary size. However, merging too many neurons will lead to an excessive output range. We need to make a trade-off between the abstraction scale and output overestimation.

**3.2 Neuron Prioritization Strategy**

The abstraction method above can merge any pair of neurons in hidden layers. The inaccuracy induced by a merging operation depends on the differences of the weights and biases of the original neurons. To get an overapproximation with lower inaccuracy, we present a heuristic strategy for prioritizing the pairs of neurons to merge.

Algorithm 1 sketches the overall process. For each pair of neurons in hidden layers, we compute a value \( m \), which takes the sum of the absolute values of the differences of corresponding incoming weights, with the addition of the absolute value of the difference between the two biases. Then we take \( m \) as the indicator and construct a min priority queue \( Q \) to guide the abstraction. When performing the abstraction, we repeatedly pop the priority queue and merge the corresponding pair of neurons in the network.

**Algorithm 1 Neuron Prioritization**

**Require:** a DNN \( D \)

**Ensure:** a min priority queue \( Q \)

```
1: \( Q \leftarrow \perp \)
2: for every pair of hidden neurons \( s_{i,b}, s_{j,b} \) do
3: \( m \leftarrow \left| b_{i,b} - b_{j,b} \right| \)
4: for every neuron \( s_{i,b} \) do
5: \( m+ \leftarrow \left| w_{i,b}^{u} - w_{j,b}^{u} \right| \)
6: end for
7: Add \((m, s_{i,b}, s_{j,b})\) to \( Q \)
8: end for
```

**3.3 Overapproximation**

We show an INN abstracted in our approach is an overapproximation of its original DNN. It implies the soundness of verifying the DNN by verifying the INN instead.

**Definition 3 (Overapproximation).** Given an INN \( A \) and a DNN \( D \), \( A \) is an overapproximation of \( D \) if and only if for any input interval \( I \), there is \( D(I, \ell) \subseteq A(I, \ell) \) for any label \( \ell \).

According to the definition, it is apparent that overapproximation is transitive for the transitivity of \( \subseteq \).

**Lemma 1 (One-step overapproximation).** Given an INN \( A \), let \( \hat{A} \) be the INN abstracted from \( A \) by merging a pair of neurons. \( \hat{A} \) is an overapproximation of \( A \).

**Proof.** Consider merging \( s_{1,1} \) and \( s_{1,2} \) into \( \hat{s}_{1,1} \). Let \( V_{i} \) denote the valuation vector of layer \( i \), \( v_{i,b} \) denote the valuation interval of the \( q \)-th neuron in layer \( i \), \( v_{i,b}^{l} \) denote the lower bound and \( v_{i,b}^{u} \) denote the upper bound.

First we prove the correctness of the first step of merging. Initially, we have \( v_{i,b}^{l} = \text{ReLU}(w_{i,b}^{l}V_{i-1} + b_{i,b}^{l}), v_{i,b}^{u} = \text{ReLU}(w_{i,b}^{u}V_{i-1} + b_{i,b}^{u}) \). After merging, \( \tilde{v}_{i,b}^{l} = \min(v_{i,b}^{l}, v_{i,b}^{l}) \), \( \tilde{v}_{i,b}^{u} = \max(v_{i,b}^{u}, v_{i,b}^{u}) \). Because \( V_{i-1} \) is non-negative and ReLU is monotonic, there are \( v_{i,b}^{l} \leq \tilde{v}_{i,b}^{l} \) and \( v_{i,b}^{u} \leq \tilde{v}_{i,b}^{u} \). Likewise, we have \( v_{i,b}^{l} \geq \tilde{v}_{i,b}^{l} \) and \( v_{i,b}^{u} \geq \tilde{v}_{i,b}^{u} \). Consequently, \( v_{i,1}^{l} \leq \tilde{v}_{i,1}^{l} \) and \( v_{i,2}^{u} \leq \tilde{v}_{i,2}^{u} \).
Then we prove the correctness of the second step. We first consider Case 1, for an arbitrary neuron \( s_{i+1,q} \), we use \( c_{i+1,q} \) and \( \hat{c}_{i+1,q} \) to denote the merged neurons’ contribution to its valuation interval, i.e., \( c_{i+1,q} = ReLU(w_{i+1,q}^l v_{i+1,1}^l + w_{i+2,q}^l v_{i+2,1}^l + b_{i+1,q}^l) \). After merging, \( \hat{c}_{i+1,q} = ReLU((w_{i+1,q}^l + w_{i+2,q}^l) v_{i+1,1}^l + b_{i+1,q}^l) \). Because \( v_{i+1,1}^l \leq v_{i+1,1}^l \) and the monotonicity of ReLU, we have \( \hat{c}_{i+1,q} \geq c_{i+1,q} \). Similarly, \( \hat{\hat{c}}_{i+1,q} \geq c_{i+1,q} \). Thus, \( c_{i+1,q} \leq \hat{c}_{i+1,q} \). Because other neurons connected to \( s_{i+1,q} \) are not altered, we have \( v_{i+1,q} \leq \hat{v}_{i+1,q} \).

We can prove that \( v_{i+1,q} \leq \hat{v}_{i+1,q} \) holds in other three cases likewise. Consequently, we have \( A(I, \ell) \subseteq \hat{A}(I, \ell) \) for any label \( \ell \) of \( A \). Thus, \( \hat{A} \) is an overapproximation of \( A \).

For the transitivity of overapproximation, it is straightforward to obtain the following theorem from Lemma 1.

Theorem 1 (Overapproximation). Given a DNN \( D \), let \( A \) be the INN abstracted from \( D \) in our approach. \( A \) is an overapproximation of \( D \).

Theorem 1 can be proved directly using Lemma 1 based on the fact that the abstraction is a finite-step process. We omit the proof due to space limitation.

4 Implementation and Evaluation

We implement our framework in Python and use a state-of-the-art MILP solver Gurobi to solve the minimization and maximization problems.

Evaluation Datasets. We consider two benchmarks, ACAS Xu networks [11] which consists of 6 hidden layers with 50 neurons in each layer and a 5 × 100 network trained on MNIST in [16]. We generate a set of random valid input regions as the evaluation dataset for the ACAS Xu network. For the MNIST network, we choose the first 100 images in the MNIST test set. Each input image can be perturbed in an \( L_\infty \) norm form ball with a bound \( \epsilon \).

Experimental Setup. All the experiments were conducted on a workstation with a 32-core 3.7 GHz AMD Ryzen Threadripper 3970X CPU and 128GB RAM. We set a timeout one hour for the verification of each input region.

Evaluation Result. Firstly, we compare the performance of our fine-grained INN abstraction to the original INN abstraction. Neuron prioritization strategy is applied to both methods to ensure they merge the same neurons. The abstractions are parameterized by \( n \times k \), where \( k \) is the number of reduced neurons in each layer and \( n \) is the number of hidden layers except the first hidden layer. We show the result of one of the ACAS Xu networks in Figure 5a and the result of the MNIST network in Figure 5b. The results of other benchmarks are similar. Our fine-grained INN has a great improvement in the precision of computation. The average output range computed by fine-grained INN is over two orders of magnitude smaller than the output range computed by original INN. With the increase of reduced neurons, the growth of the output range of fine-grained INN is much lower than that of original INN. In the cases with fewer reduced neurons, e.g., the 4 × 50 case for MNIST network, the average running time of fine-grained INN (77s) is several times longer than that of original INN (30s). However, with the increase of reduced neurons, due to the sharp fall in computation complexity, the average running time of the both become almost the same.

To show the effectiveness of neuron prioritization strategy, we compare our strategy with a random one. Both of them are based on fine-grained INN abstraction. Figure 6a shows the result. The output range of prioritization strategy is several times smaller than that of random strategy, which demonstrates our neuron prioritization strategy is very effective to improve the performance of INN abstraction.

When using neuron prioritization strategy, we have two sub-strategies to estimate the neurons. One is layer-by-layer, where we give a fixed number \( k \) as the number of reduced neurons for each layer, and select the best \( k \) pairs of neurons in each layer to be merged. The other is global, where we give a number \( j \) as the total number of reduced neurons, then select the top \( j \) pairs from all hidden layers. Figure 6b depicts the comparison between layer-by-layer strategy and global strategy, which are based on fine-grained INN abstraction for ACAS Xu networks. In our experiments, the global strategy usually performs better than layer-by-layer strategy. We find that in these cases, the
global strategy mainly merges the neurons in layer 6, 5 and 4. In few cases where layer-by-layer strategy performs better, we find the global strategy merges more neurons in layer 2 and 3 than layer-by-layer strategy. Thus we deduce that merging neurons in the front layers has more influence on the output range computation than merging neurons in the latter layers. That is because the over-estimation induced by merging is amplified layer by layer.

5 Related Work

Our work is inspired by many pioneering neural network abstraction approaches. Our approach is in line with but outperforms the abstraction approach in [13] in terms of the induced overestimation and the size of reduced neurons. Katz et al. [6] proposed an abstraction technique for merging neurons in neural networks to accelerate the verification of ACAS Xu networks. The difference is that in their approach the weights after merging are still values, unlike intervals in our approach. Another approach merges the neurons with similar behaviors, i.e., the neurons’ values are always similar for a given set of inputs [1]. This approach relies on concrete inputs of the neural networks to abstract, while our approach is independent of inputs.

Another class of abstraction-based neural network verification approaches rely on the abstraction of the constraints transformed from original neural networks, but not the abstraction of neural networks. Representative approaches include abstraction interpretation [9, 14] and linear relaxation and overapproximation [4, 19]. After abstraction, they resort to efficient linear programming solvers such as SMT and MILP solvers to check the satisfiability of abstracted constraints. Like our approach, all these approaches are sound, but refinement is needed to achieve completeness.

6 Conclusion and Future Work

We have presented a fine-grained approach to abstract neural networks for efficient formal verification. We identified four cases of merging neurons in neural networks and defined corresponding merging rules. We also introduced a neuron prioritization strategy to reduce the overestimation induced by the abstraction. Compared with the pioneering merging approach in the work [13], our approach can significantly reduce the scale of original neural networks while cause a relatively low output overestimation.

As for future work, we are planning to apply our approach to the formal verification of real-world large-scale neural networks. Further, we consider extending it to other network architectures and non-ReLU activation functions.

Acknowledgments

This work is supported by National Key Research and Development Program (2020AAA0107800), Joint Funding and AI Project (20DZ1100300) of Shanghai Science and Technology Committee, NSFC general projects (61872146, 61872144), and Open Project Fund from Shenzhen Institute of Artificial Intelligence and Robotics for Society. Weikai Miao and Min Zhang are the corresponding authors.

References

Modeling and Verification of CKB Consensus Protocol in UPPAAL

Yi-Chun Feng, Yuteng Lu, Meng Sun
School of Mathematical Sciences, Peking University, Beijing, China
{yichunfeng,luyuteng,sunm}@pku.edu.cn

Abstract—The Nervos CKB (Common Knowledge Base) is a public permissionless blockchain designed for a peer-to-peer crypto-economy network. The CKB Consensus Protocol is a key part of the Nervos CKB blockchain that improves the Consensus's performance limit of Bitcoin. In this paper, we develop a formal model of the CKB Consensus Protocol and verify some important properties of the protocol using the UPPAAL model checker. Based on the formal model, the reliability of CKB Consensus Protocol can be guaranteed.

Index Terms—CKB, Consensus Protocol, Model Checking, UPPAAL.

I. INTRODUCTION

Blockchain can be viewed as a decentralized ledger that allows direct peer-to-peer information transfer, and has numerous benefits such as persistency, decentralization and anonymity. It has been popularized since the introduction of Bitcoin by Satoshi Nakamoto [8]. Developing a trustworthy blockchain is a very challenging task due to the complexity of the distributed execution environment and the existence of vulnerabilities. In fact, a lot of attacks on blockchains succeeded in the past years, such as the famous DAO (Decentralized Autonomous Organization) attack which results in the lost of more than 50M USD [2]. Thus, many researchers conduct investigations on blockchains’ security problems.

CKB blockchain [1] has a decentralized secure layer that provides common knowledge for the peer-to-peer network. CKB Consensus Protocol [11], a key part of the CKB blockchain, is designed to overcome two Bitcoin Consensus’s drawbacks: the low transaction processing throughput and the vulnerability to selfish mining attacks. CKB Consensus Protocol limits the time of connecting the sender in search of a lost transaction. This restriction can improve transaction processing efficiency without compromising the security of the blockchain. In addition, CKB Consensus Protocol adopts the “two-step confirmation”, which is used to prevent selfish mining attacks.

In recent years, CKB has become very popular, and has been successfully applied in different areas. This makes the security properties of CKB more and more important. In this paper, we make some attempts to provide a formal framework for modeling CKB Consensus Protocol using timed automata and verifying its security properties in the UPPAAL model checker [4].

Formal verification has high assurance and coverage, so we choose it to analyze CKB Consensus Protocol’s soundness and reliability properties. In [6], UPPAAL has been used in verification of the Bitcoin Protocol, and the probability of success of double-spending attacks based on the formal model is studied. A framework for modeling the Bitcoin contracts in Promela and use SPIN to verify whether the logic of a contract is correct is provided in [3]. The interface automata model of computation is used in [7] as a semantic domain to formalize smart contracts for detecting violations of the contract agreements. The synchronization protocol is another sub-problem for the verification of the CKB blockchain. It has been discussed in [5], [10]. Based on the related works, we could see practical meaning of combining formal verification techniques and blockchain.

The main contributions of this paper are as follows:

• We propose a formal model of the CKB Consensus Protocol in which the two-step confirmation, as well as the miner, the full node, and the block propagation, are modeled using timed automata.
• We define a family of the CKB Consensus Protocol’s properties and formally verify them in the UPPAAL model checker.

The rest of this paper is organized as follows: The Nervos CKB and the CKB consensus protocol are briefly described in Section II. The formal model of the CKB consensus protocol is provided in Section III. Section IV proposes the verification of properties in UPPAAL. Section V concludes the paper and discusses possible future work.

II. A PRIMER ON CKB

Nervos Network [1] is proposed to improve scalability and provide a better user experience. It uses the idea of off-chain to create a two-layer environment. The first layer in Nervos Network is the CKB layer. It is responsible for providing decentralized and secure infrastructure. The second layer is the environment for generating states and protecting privacy. The encryption of the first layer will protect the activities in the second layer. Under the security provided by the CKB layer, the second layer’s operation can be expanded to a large extent. The operation of the Nervos Network consists of three parts: state generation executed, state verification, and storing states in the cell. After the second layer generates a new state, the state will be placed into the transaction. Later, the transactions will be broadcasted to the whole network.

The block structures in CKB include the proposal zone and the commitment zone [9], [11]. Miners put new transactions
into the proposal zone after these transactions are generated. When the transactions are put into the proposal zone, the proposal step of the two-step confirmation starts. After the proposal step, the miner will put the transaction into the commitment zone and the commitment step begins. After the two-step confirmation is completed, this transaction is considered to be “valid”. CKB protocol specifies that all blocks including orphan blocks that pass PoW will be broadcasted. The block propagation mechanism in the consensus protocol checks whether the transaction in the block is lost while avoiding extra round trips.

III. THE MODEL OF THE CKB CONSENSUS PROTOCOL

Our formal model of the CKB Consensus Protocol is mainly composed of four automata: Two-Step automaton, Miner automaton, Full Node automaton, and Block-Propagation automaton. The verification is modeled based on a single transaction subject in which not only the transaction process but also the interaction among different nodes are presented.

In this model, all variables are used to mark whether the corresponding operations are successful. The default values of all variables are 0 initially. After the operations are completed, values will be assigned according to the results. The assignment is always 1 if the operation is successful. When the operation is abnormal, the variable assignment will be greater than 1. The assigned variables are treated as parameters in the constraint conditions of state transitions.

A. Two-step Automaton

The two-step confirmation consists of the “proposal” and “commitment” step. Every transaction that passes two-step confirmation will be regarded as legal. In Fig. 1, T0 is the initial state, representing the generation of a new transaction. The mining operation is simulated by channel collectP! which is synchronous with channel collectP? in Miner automaton. Variable cp is used to mark whether the transaction has been written into the proposal zone. Variable c is the global time, which represents the time interval of each mining epoch.

If a new transaction is processed by a miner, the variable cp will be assigned in Miner automaton. T1 represents “start of proposal confirmation”. The proposal confirmation includes four operations: confirming the transaction exists in the proposal zone, checking the txid of the transaction, confirming the full node has received the transaction, and verifying the content of the transaction. The variable checkT is used to mark whether this transaction passes the txid check.

Channel checkTxid! is synchronized to checkTxid? in Full Node automaton. Variable x is the height of block on blockchain. Whenever a new block is added into blockchain, the value of x will increase by 1. Variable hp records the height of block where this transaction is located at.

In the entire two-step confirmation process, if any verification fails, the state will transfer to T9, indicating that the transaction cannot be broadcasted. State T2 means “verification of transaction”. Variables checkR and checkV indicate whether the full node successfully receives and verifies the transaction respectively. Full Node automaton will assign values to checkR and checkV after verification. State T3 means that the transaction is ready for “mining of the second step”. Variable cc is used to mark whether the transaction is written into the commitment zone. If the transaction has been verified in the first step of confirmation, it will be regarded as a “Proposed Transaction”. At the same time, the transaction will be marked as proposed at height hp if the transaction exists in one block’s proposal zone with height hp.

All transactions completed the first step of confirmation can be collected by miners and written into the commitment zone of a new block. Channel collectC! is synchronized to collectC? in Miner automaton. The difference in mining between the first and the second steps lies in the location where the transaction is written. The transaction will exist in two blocks with a certain height interval. The height interval must be limited within a defined range, and the interval will be checked in the second step of confirmation.

After the transaction is marked as proposed and written into the commitment zone, it will reach the state T4, which is “start of the commitment confirmation”. This step includes two operations: confirming the proposed transaction and checking the height interval. Variable checkC presents whether this transaction conflicts with others on the chain. Channel committed? in Full Node automaton will be synchronized. When the proposed transaction enters this confirmation, it must meet the transition constraint cc >= 1, which means that the transaction has been written in the commitment zone. The variable hc marks the current height of this block on the chain.

When the state transfers to T5, it must conform to its invariant close <= hc-hp <= far. The setting of close is to ensure the time interval is long enough for the transaction to be propagated to the entire network. The value of far is designed according to the number of proposed transactions that its device can store. If the constraint condition checkC == 1 is met, the state can transfer to T6, and the channel propagating! will be triggered at the same time. All transactions are regarded
as “Committed Transaction” when they reach state T6.

Channel propagate! is synchronized to propagate? in Block-Propagation automaton. If the transaction exists in the commitment zone of a certain block and is marked as proposed and committed, then the transaction can be spread to the network.

State T7 and T8 represent “authorization of broadcast” and “prohibition of broadcast” respectively. Variable p stands for whether this transaction can be propagated. Value 1 indicates that the transaction is legal and propagable, and value 2 indicates that the propagation will be blocked. The assignment of variable p will be completed by Block-Propagation automaton.

B. Miner Automaton

Fig. 2 demonstrates the behavior of a miner. M1 is the standby state of the miner. After mining, the state transfers to M2, which represents “new block generation”. If there is a transaction missing during block propagation, the automaton will go to channel connecting? through synchronization by Block-Propagation automaton. Channel request? and querying? describe the process of the miner being asked for the missing transaction. The miner will send the requested transaction back. Variable checkRe and checkQ represent the miner’s reply. Variable cc is the operation result after putting the transaction into the commitment zone. After two unsuccessful requests for the transaction, the miner will be regarded as a suspicious one and be blacklisted. This operation is simulated by the synchronization channel disconnecting?. State M6 indicates “disconnection”.

C. Full Node Automaton

After the new block is generated, the full node will check the legitimacy and the PoW of blocks before broadcasting them. Fig. 3 depicts the Full Node automaton in which all operations are aimed at a single transaction. In the first step of confirmation, the full node will perform the checking of transaction txid and the verification of contents, which are described by the channels checkTxid? and ReceiveVerify? respectively. In the second step of confirmation, the full node is responsible for committing the transaction. F4 is the state after the two-step confirmation automaton synchronizes the committed! channel. When any operation fails, the state transfers to F5. At this time, the transaction is deemed invalid.

D. Block Propagation Mechanism

Fig. 4 describes the simulation of block propagation mechanism. Starting from the standby state P0, the entire process is started via the synchronous channel propagating! in Two-Step automaton. The meaning of state P1 is to check whether the transaction exists in the commitment zone. Variable p indicates whether this transaction can be propagated. Value 1 means that the transaction can be broadcast, and 2 means broadcast is prohibited. When cc’s value is not equal to 1, it means that the transaction does not appear in any public blocks’ commitment zone.

State P2 represents “This transaction is previously unknown.” If the transaction is not found in the commitment zone of any public blocks, the synchronization channel connecting! must be activated to contact the miner. When checkRe >= 2, it means that the transaction is still not obtained. Then, state transfers to P3, which means “failure in request”. Block-Propagation automaton will trigger the channel querying! to synchronize to the miner, and the miner must reply within a short time t. Finally, if the missing part is still not known, the automaton transfers to state P4. State P4 represents “transaction invalidation”. When returning to the standby state, channel disconnecting! simulates the operation of blacklisting this miner and disconnecting him. At the same time, variable p is assigned to 2, which tells Two-Step automaton that this transaction should not be propagated.

IV. VERIFICATION IN UPPAAL

In this section, we provide a family of properties that should be satisfied in two-step confirmation. All the properties have been verified in the UPPAAL model checker.
Property 1. All new transactions will go through the process of being placed in the proposal zone.

\[ A \leftrightarrow \text{TwoStep.T1} \]

T1 means the transaction appears in the proposal zone. All newly generated transactions will be put into the proposal zone, then other nodes will receive the information about these transactions. The validity of these transactions will not affect the blocks’ legitimacy and the propagation’s legality.

Property 2. When a transaction is regarded as proposed, it must pass the txid check. If the transaction does not finish the txid check, it cannot become a “proposed transaction”.

\[ A [ ] \text{TwoStep.T4 imply (checkT == 1)} \]

\[ A [ ] \text{not checkT == 1 imply not TwoStep.T4} \]

Property 3. If a transaction is proposed, the full nodes must have received and verified this transaction. If the full nodes have not received this transaction or verified the contents, this transaction cannot be regarded as proposed.

\[ A [ ] \text{TwoStep.T4 imply (checkR == 1 and checkV == 1)} \]

\[ A [ ] \text{(not checkR == 1) or (not checkV == 1)} \]

\[ \text{imply not TwoStep.T4} \]

In the proposal step, the first task is processing the transaction txid, and the second task is sending a notification to the full nodes. Before a transaction is regarded as proposed, it must pass the process of checking txid (checkT == 1). If the checking is unsuccessful, it will never be considered proposed. In addition, the full nodes must have received (checkR == 1) and verified (checkV == 1) this transaction. T4 indicates that the transaction is proposed.

Property 4. When a transaction is placed in the commitment zone, it must be received and verified by the full nodes. If the full nodes have not received and verified this transaction, it will not appear in the commitment zone.

\[ A [ ] \text{TwoStep.T5 imply checkR == 1 and checkV == 1} \]

\[ A [ ] \text{not (checkR == 1 and checkV == 1)} \]

\[ \text{imply not TwoStep.T5} \]

T5 means the transaction is placed in the commitment zone. To activate the second stage of two-step confirmation, the transaction must be put into the commitment zone by miners. Before being put in the commitment zone, the transaction must pass the verification of the proposal step.

Property 5. When a transaction is committed, it must appear in the commitment zone with height hc, and satisfies the boundary: close <= hc – hp <= far.

\[ A [ ] \text{TwoStep.T6 imply checkC == 1} \]

\[ \text{and (close <= hc – hp and hc – hp <= far)} \]

T6 is a sign of transaction commitment. The value of checkC indicates whether this transaction is in the commitment zone.

Property 6. If the transaction is missing, and the miner cannot obtain the transaction after requesting and querying, the miner will be disconnected and blacklisted.

\[ A [ ] \text{BlockPropagation.P3 and BlockPropagation.P4 imply MiningNode.M6} \]

Property 7. This model is repeatable and has no deadlock.

\[ A [ ] \text{not deadlock} \]

V. Conclusion and Future Work

In this paper, we formally illustrate the complete process of the two-step confirmation of the CKB consensus protocol in UPPAAL. It may lead to a better understanding of how the CKB consensus protocol works. All the properties with clear definitions are verified formally in UPPAAL. These properties could serve as a reference for the CKB application scenarios. The notion of two-step confirmation has important implications for avoiding hiding information. This may suggest various applications except for blockchain. Having acknowledged the limitations of the scope, we offer the framework of verifying the CKB consensus protocol.

In the future, we will investigate the formal model further to determine whether the CKB consensus protocol could deal with attacks. Also, the potential of its increasing throughput needs further exploration. Additional research focusing on the application scenarios would also be of great value and interest. We are hopeful that more studies with detailed results could be provided later.

ACKNOWLEDGMENT

This work has been supported by the Guangdong Science and Technology Department (Grant no. 2018B010107004) and the National Natural Science Foundation of China under grant no. 61772038 and 61532019.

REFERENCES

Formalization and Verification of Dubbo Using CSP

Zhiru Hou, Jiaqi Yin, Huibiao Zhu*
Shanghai Key Laboratory of Trustworthy Computing,
East China Normal University, Shanghai, China

Abstract—Dubbo is a high-performance, lightweight Java Remote Procedure Call (RPC) framework developed by Alibaba, which provides interface-oriented remote method call, intelligent fault tolerance and automatic service registration. Since Dubbo is extensively applied recently as an excellent representative RPC framework, it is of great significance to formally analyze Dubbo. In this paper, we use Communicating Sequential Processes (CSP) to model and formalize Dubbo. In order to enhance the reliability of the call, we use token authentication mechanism in the modeling process. Moreover, we put the CSP description of the established model into the model checker Process Analysis Toolkit (PAT) for simulation and verification. We verify whether the four properties are valid, including Deadlock Freedom, Connectivity, Robustness and Parallelism. Our final verification results show that the model can satisfy these properties, thus we can conclude the framework can guarantee the highly available remote call.

Index Terms—Dubbo, Formalization, Verification, CSP

I. INTRODUCTION

With the development of the Internet, the architecture for a large number of website applications is constantly changing, from Monolithic Architecture, Vertical Architecture, Distributed Service Architecture to Flow Computing Architecture. Now, more and more website technicians choose to use Microservices [1], which is evolved from Service-Oriented Architecture (SOA) [3]. As a means of communication, Remote Procedure Call (RPC) [2] still plays an important role in Microservices, and Apache Dubbo is an excellent representative of the RPC framework.

Dubbo [4] is an open source and high-performance RPC call framework developed by Alibaba. It is a RPC remote call service solution dedicated to providing high performance and transparency. In recent years, some work has been done on Dubbo [5, 6]. Zhang et al. [5] proposed a distribution network state control system using Dubbo in order to improve the lean management level of the distribution network. Xiong et al. [6] designed a new type of think tank evaluation system based on Microservices, and realized the communication between services based on the RPC remote call of Dubbo distributed framework. From these works, we can find that they focused more on using Dubbo to implement remote calls between services. Unfortunately, there is nearly no research conducted to describe the interactions in Dubbo formally, thus it is a challenge to give a formal model on the interactions between the components in Dubbo.

In this paper, we propose a formal model of Dubbo using Communicating Sequential Processes (CSP) [7], which aims to reflect the interactions of Dubbo’s call process. In order to better ensure the reliability of calling services, token authorization mechanism is also formalized in this model. In addition, we use Process Analysis Toolkit (PAT) [8, 11] to verify whether the achieved model caters for some significant properties or not, including Deadlock Freedom, Connectivity, Robustness and Parallelism.

The remainder of this paper is organized as follows. Section II gives a brief introduction to Dubbo and the process algebra CSP. In Section III, we formalize the model of Dubbo using CSP. Furthermore, in Section IV, we apply the model checker PAT to implement the achieved model and verify four properties. Finally, we give a conclusion and make a discussion on the future work in Section V.

II. BACKGROUND

In this section, we give a brief introduction to Dubbo’s call service process, token authentication and process algebra CSP.

A. Dubbo

Dubbo is a distributed service framework. The architecture of Dubbo is shown in Fig. 1. As we have seen in Fig. 1, Dubbo architecture mainly has four components, including provider, consumer, registry and monitor. Furthermore, Fig. 1 shows the main communications of Dubbo architecture, and their respective functionalities are seen in Table I.

<table>
<thead>
<tr>
<th>Components</th>
<th>Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provider</td>
<td>Exposing remote services</td>
</tr>
<tr>
<td>Consumer</td>
<td>Calling the remote services</td>
</tr>
<tr>
<td>Registry</td>
<td>Service discovery and configuration</td>
</tr>
<tr>
<td>Monitor</td>
<td>Counting the number of service invocations and time-consuming</td>
</tr>
<tr>
<td>Container</td>
<td>Managing the services’ lifetime</td>
</tr>
</tbody>
</table>

In Fig. 1, when consumer wants to call the service it needs, the following sequence of actions occurs:

1. Container is responsible for launching, loading and running the provider.
2. Provider registers its services to registry when it starts.
3. Consumer subscribes the needed services from the registry when it starts.
4. Registry returns a list of providers to consumer. When the list changes, the registry will push the changed data to consumer through long connection.

DOI reference number: 10.18293/SEKE2021-077
(5) Consumer selects one of the providers based on load balancing algorithm and executes the invocation. If fails, it will choose another provider.

(6) When monitor starts, it will subscribe all providers and consumers that registered or called.

(7) Both consumer and provider count the number of service invocations and time-consuming in memory, and send the statistics to monitor every minute.

In Dubbo, if the provider wants to verify the identity of the consumer before the consumer invokes its service, the system can use token authentication between them. In this condition, the consumer cannot bypass the registry and connect directly to provider. The details of using token authentication in Dubbo can be seen in Fig. 2.

There are two types of tokens in Dubbo, which are random token and fixed token. Random token is generated using a UUID, and fixed token is equivalent to the password which is used in this paper. The sequence of using token verification [12] in Dubbo is:

(1) When provider registers its service, it generates a token and publishes it with the service to registry.

(2) Registry has the right to decide whether to assign token to consumer.

(3) Once the consumer obtains the URL of the provider from registry, it can request to invoke the provider through the token.

(4) The provider needs to verify whether this token is consistent with the token generated by itself. If it is non-consistent, this invocation will fail.

B. A Brief Introduction to CSP

CSP was proposed by C.A.R Hoare, which is the abbreviation of Communicating Sequential Processes [7]. It has been successfully applied to model and verify diverse concurrent systems and protocols [9, 10]. We use the following syntax to define the processes in this paper.

\[ P, Q ::= \text{SKIP} \mid \text{STOP} \mid a \rightarrow P \mid c?x \rightarrow P \mid cle \rightarrow P \mid P<|b>|Q \mid P\parallel Q \mid P||Q \mid P;Q \]

- \text{SKIP} stands for a process which terminates successfully.
- \text{STOP} represents that the process does nothing and runs into the deadlock state.
- \( a \rightarrow P \) performs action \( a \) firstly, then behaves like \( P \).
- \( c?x \rightarrow P \) receives a message by channel \( c \) and assigns the received message to variable \( x \), then behaves like \( P \) subsequently.
- \( cle \rightarrow P \) sends a message \( e \) through channel \( c \), then the subsequent behavior is \( P \).
- \( P<|b>|Q \) represents a conditional choice. If the expression \( b \) is true, process \( P \) will be carried out; otherwise, process \( Q \) is executed.
- \( P\parallel Q \) is a general choice, it acts like either \( P \) or \( Q \) and the environment decides the selection.
- \( P||Q \) shows the parallel composition between \( P \) and \( Q \). The \( || \) means that actions in the alphabet of both operands require simultaneous participation of them.
- \( P||Q \) indicates that \( P \) interleaves \( Q \) which means \( P \) and \( Q \) run concurrently without barrier synchronization.
- \( P;Q \) executes process \( P \) and process \( Q \) in sequence.

III. MODELING DUBBO

In this section, we give a formal model of Dubbo’s call service process, and this model includes five components. The formalization proceeds based on the four components described in Section II. In order to better describe the temporal process of Dubbo, we propose a new component \( \text{Clock} \).

A. Overall Modeling

For the whole system, there are four crucial processes running in parallel through their own corresponding channels, including \text{Provider}, \text{Consumer}, \text{Registry} and \text{Clock}. \text{Monitor} process interleaves with them. The behavior of Dubbo system process is modelled as below.

\[ \text{DubSys} =_df \text{Provider} \parallel \text{Consumer} \parallel \text{Registry} \parallel \text{Clock}; \ 
\text{System} =_df \text{DubSys} \parallel \text{Monitor}; \]

Next, we give the formalization of \text{Provider}, \text{Consumer}, \text{Registry}, \text{Monitor} and \text{Clock}, respectively.
ServProvider registers its services and token to illustrated in Table II.

In this system, there can be several providers. Each provider has a unique ID marked as $i$, and $I$ is the total number about providers. Provider is mainly responsible for providing services and generating tokens. In addition, Provider periodically sends a heartbeat to registry and a monitor message to monitor. Thus, we formalize Provider as below.

**Provider**

Before introducing the three processes of Provider, we first explain messages and channels used here. The messages can be described as follows, and the explanations of channels are illustrated in Table II.

- **ProListInfo** is sent from Provider to Registry, which contains the ID, IP address, host name and the corresponding information of the Provider.
- **InvokeSuccess** is a reply from Provider to Consumer, which means that Consumer can call the matched Provider successfully.
- **InvokeFail** is a reply from Provider to Consumer, which represents that the invocation fails.
- **TokenFail** is a reply from Provider to Consumer, which means that the token sent by Consumer can call the provider within timeout seconds, the call is successful; otherwise it fails. The detailed behavior is modelled as follows.

**ServProvider**

ServProvider describes the details of publishing services and being called by consumers. At first, Provider registers its services and token to Registry. When Provider receives the call request from Consumer, it first verifies whether the token provided by consumer matches the token generated by itself. If the match is successful, the authentication is passed; otherwise, the authentication fails and TokenFail is sent to Consumer. In addition, when the monitor starts, Provider sends ProListInfo to Monitor asynchronously. The behavior of ServProvider is modelled as below.

**ServProvider**

The following is the model of the WAIT function, where the parameter $t$ is the unit of time to wait, and the specific model is as follows.

```
WAIT(t) = SKIP((t == 0) && (tick=WAIT(t - 1)))
```

**ServPMon.** ServPMon process is mainly used to send monitor messages to monitor regularly. Once the monitor starts, it asks the current time and waits MonInterval seconds. Then it sends MonPro to Monitor and cycles continuously. Next we give the formalization of ServPMon.

```
ServPMon = SKIP TimeRequest -> TimeRequest MonInterval -> WAIT(MonInterval) ProListInfo 
```

<table>
<thead>
<tr>
<th>Channels</th>
<th>Functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_iR$</td>
<td>Transmitting register messages between providers and registry</td>
</tr>
<tr>
<td>$C_dR$</td>
<td>Transmitting subscribe messages between consumers and registry</td>
</tr>
<tr>
<td>$P_iC_d$</td>
<td>Transmitting call messages between consumers and providers</td>
</tr>
<tr>
<td>$C_dM$</td>
<td>Transmitting consumers’ monitor messages between consumers and monitor</td>
</tr>
<tr>
<td>$P_iM$</td>
<td>Transmitting providers’ monitor messages between providers and monitor</td>
</tr>
<tr>
<td>$ConHeart_i$</td>
<td>Transmitting heartbeat messages between providers and registry</td>
</tr>
<tr>
<td>$Time$</td>
<td>Transmitting time messages</td>
</tr>
</tbody>
</table>

**B. Provider**

**TABLE II**

**THE EXPLANATIONS OF CHANNELS OF THE MODEL**

- **Provider**
- **ServPMon**
- **ServHBeat**
- **ServListInfo**
- **ServListMon**
- **ServListReg**

For the process InvkProvider, since the authentication passes, it is necessary to check whether the provider is occupied by other services. If the provider is not occupied, Provider sends InvokeSuccess to Consumer. It also increases the number of service invocations and calculates time-consuming using the process Clock; otherwise, the consumer can wait timeout seconds. Suppose consumer can call the provider within timeout seconds, the call is successful; otherwise it fails. The detailed behavior is modelled as follows.

```
InvkProvider = df

P_iC_dInvokeSuccess(OccupiedState_i,d = true) ->
Add(PListCount, PCount, Time) ->
TimeRequest PStart := t ->
P_iC_dEnd(OccupiedState_i, d = false) ->
TimeRequest TimeRequest PEnd := t ->
End PTime_i := PEnd - PStart ->
Calc(MonPro ? PCount, PTime_i) ->
ServProvider
```

The following is the model of the WAIT function, where the parameter $t$ is the unit of time to wait, and the specific model is as follows.

```
WAIT(t) = SKIP ((t == 0) && (tick=WAIT(t - 1)))
```

**ServPMon.** ServPMon process is mainly used to send monitor messages to monitor regularly. Once the monitor starts, it asks the current time and waits MonInterval seconds. Then it sends MonPro to Monitor and cycles continuously. Next we give the formalization of ServPMon.

```
ServPMon = SKIP TimeRequest -> TimeRequest MonInterval ->
WAIT(MonInterval) ProListInfo 
```

- **MonPro** is transmitted from Provider to Monitor, which owns the ID, the number of service invocations and time-consuming of the Provider.
- **request** is used by asking Clock the current time.
- **HeartBeat** is sent from Provider to Registry, which indicates the Provider is still running.
- **ProListInfo** is transferred from Provider to Monitor, which contains the URL addresses of all providers.
**ServHBeat.** ServHBeat works in heartbeat mechanism, which means that Provider needs to send a heartbeat to Registry regularly. Then we formalize the process of ServHBeat as below.

\[
\text{ServHBeat} = \delta t \text{TimeRequest} \to \text{TimeStart} \to \begin{cases} 
\text{ComHeartBeatHeartBeat} \\
\langle \text{StartLast := Start} \rangle \text{ServHBeat}
\end{cases}
\]

Provider asks Clock for the current time firstly. If the time interval is less than HBeatInterval, Provider sends a request to Clock again; otherwise, Provider sends HeartBeat to Registry directly and this process cycles continuously.

**C. Consumer**

Like Provider, each consumer has a unique ID marked as \(d\), and \(D\) is the total number about consumers. Consumer mainly expresses subscribing service and calling service. Moreover, Consumer sends a monitor message to monitor regularly. Thus, we formalize Consumer as below.

\[
\text{Consumer} = \delta t \mid \mid \mid \text{ServConsumer} \mid \mid \mid \text{ServCMon}
\]

The messages in Consumer can be described as follows.

- **SusRe** is sent from Consumer to Registry, which contains the ID, IP address and the corresponding information of the Consumer.
- **InvoRe** is transmitted from Consumer to Provider, including the IDs of Consumer and Provider together with invocation request.
- **MonCon** is sent from Consumer to Monitor, which contains the ID, the number of service invocations and time-consuming of the Consumer.
- end is transmitted from Consumer to Provider, which means that Consumer wants to finish the call process.
- **ConListInfo** is transferred from Consumer to Monitor, which owns the URL addresses of all consumers.

**ServConsumer.** ServConsumer focuses more on subscribing services and initiating the call processes. After Consumer sends subscription to Registry, Consumer can attain a list of providers and the tokens from Registry. Then Consumer verifies whether the states of providers are available or not. Consumer can select an available provider to call via load balancing algorithm. Moreover we use Random Load Balance algorithm here, which is selected according to the provider’s weight and sets a random probability. Consumer sends invocation request to Provider and waits the reply. In addition, if ProList changes, Registry will notify Consumer asynchronously. Once monitor starts, Consumer needs to send ConListInfo to Monitor. After the above analysis, ServConsumer is formalized as below.

\[
\text{ServConsumer} = \delta t \begin{cases} 
\text{CdR!SusRe} \to \text{CdR?ProList} \to \\
\langle \text{State] == Open} \rangle \to \text{SkIP} \\
\langle \text{CdR?ModiProList} \to \text{SkIP} \\
\langle \text{CdM!StartM} \to \text{CdM!ConListInfo} \to \text{ServConsumer}
\end{cases}
\]

For InvkConsumer, if reply is InvokeSuccess, it calculates time and increases the number of invocations as Provider; by contrast, it can have two opportunities to try to call other providers. Then, we formalize the process InvkConsumer as below.

\[
\text{InvkConsumer} = \delta t \begin{cases} 
\text{Initial} \{\text{CCount} = 0; \text{RandLoadBan}(PID)\} \\
\text{CdM!InvoRe} \cdot \text{CToken} \to \text{CdM?reply} \to \\
\langle \text{Add(CCount)}; \text{TimeRequest} \to \\
\text{Time!=t} \to \text{CdM?call} \to \\
\text{TimeRequest} \to \text{Time!=t} \to \text{CdM?End} \to \\
\text{End(CTime, EEnd, CStart)} \to \\
\text{Calc} (\text{MonCon} \cdot \text{CCount} \cdot \text{CTime}, \text{C}) \\
\text{ServConsumer} \\
\langle \text{reply == InvokeSuccess} \} \\
\{x > 0 \\
\{x = 0 \}
\end{cases}
\]

**ServCMon.** ServCMon process is mainly used by consumer to send monitor messages to monitor regularly. Once the monitor starts, Provider needs to send MonCon to Monitor. Like ServPMon, we give the formalization of ServCMon.

\[
\text{ServCMon} = \delta t \text{TimeRequest} \to \text{TimeStart} \to \\
\text{Wait(MonInterval); CdM!MonCon} \to \text{ServCMon}
\]

**D. Registry**

We use Zookeeper [13] to implement dynamic registration and discovery of services in the registry. Registry serves as a component for storing information and receiving the heartbeat message from providers. Thus, we formalize Registry as below.

\[
\text{Registry} = \delta t \text{ServRegistry} \mid \mid \text{RegHBeat}
\]

The messages in Registry can be described as follows, and the channels are explained in Table II.

- **ProList** is sent from Registry to Consumer, and it is a list which contains matching providers’ information.
• ModifProList is transferred from Registry to Consumer, which owns modified matching providers’ information.

Next, we formalize the two processes, respectively.

**ServRegistry.** ServRegistry process is applied for describing the registration and subscription processes. Firstly Registry receives registration from Provider and subscription from Consumer, respectively. Based on the information provided by Consumer, Registry checks whether there is a matching provider. If there is no matching provider, then it skips; otherwise, Registry finds out the relevant providers according to the matching algorithm SelectPro, and sends ProList to Consumer. The behavior of ServRegistry process is modelled as below.

\[
\text{ServRegistry} =_{def} \text{Initial}\{\text{ProList} = \text{null}\}; \\
P, R?\text{ProInfo.Token} \rightarrow C_d?\text{SusRe} \rightarrow \\
(\text{SelectPro}(\text{ProList} \setminus \text{ProInfo.IP.Token}); \\
C_d?\text{ProList} \rightarrow \text{ServRegistry} \\
\angle(\text{SusRe}.\text{CInfo} \in \text{ProInfo.PSer}) > \text{SKIP})
\]

**RegHBeat.** RegHBeat process mainly involves the heartbeat mechanism. The process RegHBeat is formalized as follows.

\[
\text{RegHBeat} =_{def} \\
(\text{ComHeart},?=\text{HeartBeat} \rightarrow \text{RegHBeat}) \\
\text{Initial}\{\text{ModifProList} = \text{ProList}\}; \\
\square(\text{state}_i = \text{closed}); \\
\text{Modif}\{\text{ModifProList} \setminus \text{ProInfo.IP}\}; \\
C_d?\text{ModifProList} \rightarrow \text{SKIP}
\]

In case Registry receives heartbeat message from Provider, it indicates the provider is running normally; on the other hand, it means that the provider may be down, and we can modify the provider’s information to ModifProList.

**E. Monitor**

Monitor is responsible for monitoring the status of the service. Thus, Monitor can be formalized as below.

\[
\text{Monitor} =_{def} P, M!\text{StartM} \rightarrow P, M?\text{ProListInfo} \rightarrow \\
C_d!\text{StartM} \rightarrow C_d?\text{ConListInfo} \rightarrow \\
P, M?\text{MonPon} \rightarrow C_d?\text{MonCon} \rightarrow \text{Monitor}
\]

When monitor starts, it needs to obtain the URL information of all providers and consumers. It also receives MonPro from Provider and MonCon from Consumer, respectively.

**F. Clock**

In order to better represent the temporal process of Dubbo, we abstract Clock process, which is used to express the global clock. Once other processes ask Clock for the time via the channel Time, Clock will send back the current time \(t\) which is a positive integer. The processes of \(\text{Clock}(t)\) can be described as follows.

\[
\text{Clock}(t) =_{def} (\text{tick} \rightarrow \text{Clock}(t + 1)) \\
\square(\text{Time!request} \rightarrow \text{Time!t} \rightarrow \text{Clock}(t))
\]

**IV. Verification**

In this section, we implement CSP model mentioned in Section III and verify some important properties using PAT.

**A. Verification in PAT**

Before verifying the properties, we define some significant variables. \(I, D, R, M\) denote the number of the providers, the consumers, the registry and the monitor. In the trial, we set \(I, D, R, M\) to be 2, 3, 1, 1, respectively.

**Property 1: Deadlock Freedom**

In Dubbo, we should avoid the situation that two or more consumers are waiting the resources which have been occupied by other consumers infinitely. In addition, System\(1()\) should also meet Deadlock Freedom. For the explanation of System\(1()\), see Property 2. In the tool PAT, there is a primitive to describe this situation:

\[
\text{assert System()} \text{deadlockfree}; \\
\text{assert System1()} \text{deadlockfree};
\]

**Property 2: Connectivity**

Registery and monitor are optional, and consumer can connect provider directly in Dubbo. However, we use token to enhance identity authentication in this paper, so that consumers need to go through registry to connect with the provider. Thus we prove that monitor is optional here.

We hide the relevant channels of monitor to detect whether the provider can successfully connect with consumer without monitor, we use System\(1()\) to model this in PAT. If the monitor is optimal, the variable CncStatePro and CncStateCon should be True. Moreover, both System\(()\) and System\(1()\) should satisfy this property. The assertion about this property is defined as below:

\[
\text{System1()} = \text{System()} \setminus \{P, M, C_d\}; \\
\text{define Connectivity(CncStatePro} == \text{true } \\
\&\& \text{CncStateCon} == \text{true});
\]

**Property 3: Robustness**

The primary objective of Dubbo is to accomplish the call of provider reliably even in the presence of failures. If providers are stateless, one instance’s downtime does not affect the usage. After all the providers of one service go down, consumer infinitely reconnects to wait for service provider to recover.

In this paper, we assume that the services called by consumers are the same as those provided by providers. Here we define that there are four valid conditions listed as follows. The first and second conditions are that all providers can run normally, the third condition is that the first provider is down
and the last condition is that the second provider is down. The assertion is defined as below:

```c
#define Robust1(PCount[0] == 1 && PCount[1] == 2);
#define Robust2(PCount[0] == 2 && PCount[1] == 1);
#define Robust3(PCount[0] == 0 && PCount[1] == 3);
#define Robust4(PCount[0] == 3 && PCount[1] == 0);
#define Robustness(Robust1 && Robust2)
|| Robust3 && Robust4);
#define Parallelism(Paral && Par2);
#define System() reaches Parallelism;
```

**Property 4: Parallelism**

Parallelism means that the system allows multiple providers publish services and consumers subscribe services concurrently, the processes do not interfere with each other. We define two Boolean variables, `aplPro` means the number of registration submissions of providers, and `aplCon` means the number of subscription submissions of consumers. Our goal is that the system can reach a state where the value of `aplCon` and `aplPro` should be 1, which reflects the providers and the consumers can involve calling processes parallelly. The assertion about this property is defined as below:

```c
#define Paral1(aplPro[0] == 1 && aplPro[1] == 1)
#define Paral2(aplCon[0] == 1 && aplCon[1] == 1
&& aplCon[2] == 1);
#define Parallelism(Paral && Par2);
#define System() reaches Parallelism;
```

### B. Verification Results

The verification results are showed in Fig. 3. From Fig. 3, we can easily find that the four properties are all valid, which represents that the constructed model caters for the specifications and these properties.

1) Deadlock Freedom means that the constructed model does not run into a deadlock state.
2) Connectivity is valid which means that the provider and the consumer can connect successfully, even without monitor.
3) Robustness represents that the framework has good fault tolerances, which is an important property for RPC framework.
4) Parallelism indicates that the providers can commit registrations and the consumers can commit subscriptions concurrently.

**V. CONCLUSION AND FUTURE WORK**

Dubbo is a high-performance distributed service framework from Alibaba, which can provide good remote call. In this paper we analyzed Dubbo and used token mechanism to enhance identity authentication. We applied process algebra CSP in formalizing Dubbo. Subsequently, we used PAT to encode the CSP description and verified this model. In addition, we performed the validation of four properties, including Deadlock Freedom, Connectivity, Robustness and Parallelism. These properties are all valid. Therefore, we can conclude that our model satisfies these properties and the framework can realize effective remote calls from the perspective of process algebra.

The formal verification of the distributed service framework is still a challenge. In the future, we will formalize and verify the Dubbo with Zookeeper [13] in more details and verify whether the framework can resist attacks or not.

**Acknowledgements.** This work was partly supported by National Key Research and Development Program of China (Grant No. 2018YFB2101300), National Natural Science Foundation of China (Grant No. 61872145, 62032024), Shanghai Collaborative Innovation Center of Trustworthy Software for Internet of Things (Grant No. ZF1213).

**REFERENCES**

Conv-Reluplex : A Verification Framework For Convolution Neural Networks

Jin Xu†, Zishan Li*, Miaomiao Zhang*, Bowen Du†

*School of Software Engineering, Tongji University, Shanghai, China
†Department of Computer Science, University of Warwick, Coventry, United Kingdom

Email: miaomiao@tongji.edu.cn; B.Du@warwick.ac.uk

Abstract—In recent years, machine learning has demonstrated impressive performance in many real-world tasks, especially in computer vision and natural language processing. However, to apply them in safety-critical systems one needs formal guarantees on the neural network outputs. The Reluplex tool is proposed to verify the safety of deep neural networks (DNNs), and in case the DNN fails to give a correct output, can generate adversarial examples. Since the tool can only handle DNNs, it is necessary to extend the tool to process image data. Therefore, in this paper, we propose the Conv-Reluplex framework, which is designed to verify the convolutional layer and pooling layer in convolutional neural networks (CNNs), and generate adversarial examples when classification is misguided. We conduct several experiments on MNIST to evaluate our approaches. The results show that the original CNN is improved using the adversarial examples generated by our tool, and the precision of classification can be increased significantly.

Index Terms—Reluplex Algorithm, Adversarial Robustness, Verification Framework

I. INTRODUCTION

In recent years, machine learning [1] [2] has been widely used in various fields, such as image recognition [3], speech recognition [4] and autonomous vehicles [5]. Neural networks are trained over a finite training set and are expected to generalize, i.e., to behave correctly for previously-unseen inputs. However, Szegedy et al. discovered that the input-output mapping learned by neural networks is discontinuous to a large extent [6]. It turns out that perturbed inputs similar to a correctly classified input could be misclassified by deep learning models with high confidence, which are generally called adversarial examples [6]. There is an urgent need for methods that can provide formal guarantees about neural networks behavior. Unfortunately, manual reasoning about large neural networks is impossible, as their structure renders them incomprehensible to humans. Automatic verification techniques are thus needed.

Verification of neural networks is difficult as it is experimentally beyond the reach of general-purpose tools such as linear program (LP) solvers or existing satisfiability modulo theories (SMT) solvers [7]–[9]. In [9], the authors propose an approach for verifying the local adversarial robustness of DNNs based on a systematic exploration of a region. The verification process is still exponential in the number of features. Katz et al. propose an algorithm called Reluplex, which is efficient to verify DNNs with ReLU activation functions [12]. This is achieved by leveraging the piecewise linear nature of ReLUs and attempting to gradually satisfy the constraints that they impose as the algorithm searches for a feasible solution. We call the algorithm Reluplex, for “ReLU with Simplex”. Compared with [9] and [10], Reluplex can handle larger deep neural networks and guarantee that there are no irregularities hiding between the discrete points.

Reluplex mainly focuses on DNNs. To support image processing, it is necessary to extend this tool with CNNs to make it more practical. To this end, we present a Conv-Reluplex framework on the basis of Reluplex. The extension is able to verify network robustness, that is, if Conv-Reluplex finds that the adversarial robustness is not satisfied, a corresponding counter-example (adversarial example) will be generated in the form of an image type which shows an abnormal classification with respect to the network. So, in the Conv-Reluplex framework, the adversarial examples generation is conducted during verification process, not alike those in the existing methods [13]–[15]. As have been shown in [10] [14] [16] [17], adversarial training can improve the robustness of models. We thus make some tricks in the tool to produce a large number of adversarial examples, not just one, for our later training of CNNs. The experimental results show that appropriate adversarial training is helpful to enhance the adversarial robustness of the CNNs.

So, based on the current tool—Reluplex, we give the Conv-Reluplex framework to verify CNNs, meanwhile generate adversarial examples. We also have implemented the framework [18] and conducted some experiments. In the following, we begin with some background on CNNs, and Reluplex in Section 2. In Section 3, we present Conv-Reluplex verification framework, also with an emphasis on generation of adversarial examples, followed by experimental results and analysis in Section 4. We conclude the paper in the last section.

DOI reference number: 10.18293/SEKE2021-085
II. Preliminaries

We first recall some definitions of CNNs, Adversarial robustness and Reluplex algorithm.

A. Convolutional Neural Networks

A convolutional neural network [19] is comprised of one or more convolutional layers (often with subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of CNNs is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units.

B. Adversarial Robustness

Adversarial robustness [12] is a safety property, which measures the resilience of a neural network against the inputs with perturbation. When the input is $x_0$, the output of network can be denoted as $f(x_0)$. A neural network is $\delta$-locally-robust at point $x_0$ iff

$$\forall x, \quad \|x - x_0\| \leq \delta \Rightarrow f(x) = f(x_0)$$

Intuitively, the above formula states that for input $x$ that is very close to $x_0$, the network assigns to $x$ the same label that it assigns to $x_0$; “local” thus refers to a local neighborhood around $x_0$. Larger values of $\delta$ imply larger neighborhoods, and hence better robustness [16].

C. Reluplex Algorithm

Reluplex is an SMT solver for a theory of linear real arithmetic with ReLU constraints. The technique is based on extending the Simplex algorithm [17] [18] to support the non-convex ReLUs in a way that allows their inputs and outputs to be temporarily inconsistent which will then be fixed as the algorithm progresses. In other words, DNNs and their properties can be directly encoded as conjunctions of linear formulas and ReLU constraints. Here, a ReLU constraint satisfies $x^f = \max(0, x^b)$, where $x^f$ and $x^b$ stand for the connection information of the nodes. To guarantee termination, some ReLU connections may need to be split upon. However, in many cases this is not required, resulting in a practically efficient solution. The details that are crucial to performance and scalability, such as the use of floating-point arithmetic, bound derivation for ReLU variables, and conflict analysis, are discussed in [12]. The success in verifying properties of the ACAS Xu networks [20] indicates that the technique holds potential for verifying real-world DNNs.

III. A VERIFICATION FRAMEWORK FOR CNNS BY CONV-RELUPLEX

The Reluplex is able to verify and improve the adversarial robustness of networks. If the adversarial robustness is not satisfied for a classification task, there must exist an $x'$, $\|x' - x\| \leq \delta$, with $x'$ belonging to a label different from that of $x$. We call $x'$ a counter-example, which shows the violation of the adversarial robustness property. As the tool is not able to directly verify the CNNs and generate their corresponding counter-examples, we hereby propose the ConvReluplex framework to boost its ability on CNNs.

CNNs generally consist of three kinds of layers, convolutional layers, pooling layers and fully connected layers. The convolutional layer extracts rough features from the original image, the pooling layer is a form of non-linear down-sampling to generate dominant features, and the fully connected layer employs these dominant features for classification. The fully connected layer closely resembles the basic component of DNNs, so the Reluplex can be directly applied to process this layer. However, to tackle convolutional and pooling layers in the CNNs, in particular for the purpose of adversarial example generation, we propose a reverse calculation algorithm.

As illustrated in Fig. 1, the designed Conv-Reluplex framework consists of Reluplex and a reverse calculation algorithm. Given a trained CNN classifier, there are six steps included in the framework.

- Step 1: A correctly classified image sample is selected from the training dataset and input into the CNN.
- Step 2: When the image sample passes through the convolutional layer and the pooling layer, the rough features and the dominant features extracted by these two layers are stored respectively before it passes through the fully connected layer.
- Step 3: The stored dominant features are regarded as the input of a DNN, because the structure of the DNN is almost same as the fully connected layer. A small perturbation neighbourhood $\delta$ of the stored dominant features is a set, and Reluplex is used to verify the adversarial robustness of the fully connected layer.
- Step 4: The adversarial robustness of the DNN (the fully connected layer) is verified from Reluplex. If it is not satisfied, Reluplex will output a counter-example to prove that the adversarial robustness is not satisfied. This example is called as ”intermediate adversarial example”.
- Step 5: The Unpooling algorithm (See Sec.III.A) is applied to restore the intermediate adversarial example to the potential rough features before the pooling operation.
- Step 6: The Deconvolution algorithm (See Sec.III.B) is applied to restore the potential rough features to a image.

Reluplex is able to find out a counter-example (the intermediate adversarial example in our framework), in general, Conv-reluplex can turn this intermediate adversarial example into a real image.

A. Unpooling algorithm

The function of pooling layers is to reduce the dimension of rough features and generate dominant features. The commonly used types of pooling calculation are Max, Average, Sum, etc. The Max-pooling is shown in Fig. 2. The input of this pooling layer is a $4 \times 4$ matrix , the pooling window is a $2 \times 2$
matrix, and the stride is 2. The first iteration operation of the pooling layer is $\text{Max}(1, 1, 5, 6) = 6$, then sliding to the right of 2 cells, the second pooling operation is $\text{Max}(2, 3, 4, 5) = 5$, and so on. The final output of the pooling layer is a 2*2 matrix.

Fig. 2: Max-pooling calculation

The Unpooling algorithm is the inverse calculation of pooling operation. The potential rough features can be generated based on the Unpooling algorithm using the intermediate adversarial example found in Step 4 and the original rough features stored in Step 3. Because the CNN is well-trained, the internal structure and weight parameters is fixed. Our approach is to make minimal changes to the original rough features so that it match the intermediate adversarial example.

For the Max-pooling, our Unpooling algorithm goes through every value in the pooling window. If the value is greater than the corresponding value in the intermediate adversarial example, the value is set as the corresponding value. If the value is less than or equal to its corresponding value, the value will remain. Assume that $x$ is the value in the original rough features, and $\alpha$ is its corresponding value in the intermediate adversarial example, the Unpooling algorithm is defined as following:

$$f_{\text{unpooling}}(x) = \begin{cases} x = \alpha, & \text{if } x > \alpha \\ x = x, & \text{if } x \leq \alpha \end{cases}$$

An example of the Unpooling algorithm for the Max-pooling is illustrated as Fig. 3. The left matrix is the original rough features, the middle matrix is the intermediate adversarial example. Firstly, the values 1,1,5,6 are in the window of the first pooling operation, and compared with the corresponding value in the intermediate adversarial example 4. Secondly, because 5 and 6 are larger than 4, so the value of 5 and 6 reduce to 4, and other two values, 1 and 1, are both smaller than 4, so they remain. The pooling window slides to the next position iteratively. Finally, the new input matrix on the right side of Fig. 3 is generated.

Fig. 3: Unpooling calculation of Max-pooling

As mentioned before, the Unpooling algorithm only modifies the value necessarily. Therefore, the minimal changes can be ensured.

B. Deconvolution algorithm

The function of convolutional layers in CNNs is to extract the rough features from the original image, and the operation is demonstrated in Fig. 4. The input of the convolutional layer is a 5*5 matrix, the kernel (parameter) of this layer is a 3*3 matrix, and the stride is 1. When the third iteration of convolution is performed, The kernel slides to the upper right corner of the input. The corresponding values in the same location of these two matrices multiply with each other and the values of iterations are summed up. The result is shown at the upper right corner of the output matrix. When the operations of convolution are completed, the output of the convolutional layers, a 3*3 matrix, is obtained.

Fig. 4: Convolution calculation

Generally, a nonlinear activation function follows the convolutional layer. ReLU function is a widely used nonlinear activation function after the convolutional layer, and it can be expressed as follows.

$$Y_i = \text{ReLU} (X_i \cdot W_i + B_i)$$
where \( X_i \) represents input data of layer \( i \), \( W_i \) represents convolution kernel parameter of layer \( i \), \( B_i \) represents bias of layer \( i \), and \( Y_i \) represents feature map of layer \( i \).

In order to convert the rough features back to the image which violate the adversarial robustness, the deconvolution algorithm should invert both the ReLU operation and the convolution operation. The problem of convolution layer inversion is formulated as a linear constraint solving problem. The kernel \( W_i \), the bias \( B_i \) and the potential rough features \( Y_i \) is obtained from Step 2 and Step 5 respectively. \( X_i \) is the adversarial image to be generated. Since the result of the ReLU function is known, according to \( Y_i \) and the ReLU definition, the ReLU constraint can be eliminated from the expression, and the problem is directly encoded as the following constraints:

\[
Y_i = \text{ReLU} (X_i \cdot W_i + B_i) \iff \begin{cases} 
X_i \cdot W_i + B_i = Y_i, & \text{if } Y_i > 0 \\
X_i \cdot W_i + B_i \leq 0, & \text{if } Y_i = 0
\end{cases}
\]

For the first convolution layer, and the input of this layer is still the input of the CNN, an additional upper and lower bound constraint of \( 0 \leq X_i \leq 1 \) needs to be added for each variable \( X_i \). Because in preprocessing stage, the pixel values of 0 to 255 are usually normalised between 0 to 1, the additional constraints ensure that the generated data obtained by the inversion can be correctly converted into a image.

\[
\begin{array}{cccccccc}
X_1 & X_2 & X_3 & X_4 & X_5 & X_6 & X_7 & X_8 & X_9 \\
6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 \\
X_{15} & X_{16} & X_{17} & X_{18} & X_{19} & X_{20} & X_{21} & X_{22} & X_{23} \\
\hline
1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\
5 & 5 & 4 & & & & & & \\
\end{array}
\]

![Fig. 5: Deconvolution calculation](image)

An example of the Deconvolution algorithm is shown in Fig. 5. The input data (the normalised image) to be generated is a 3×5 matrix, and all values, \( x_1,..,x_{15} \), in the matrix are unknown. The convolution kernel is a 3×3 matrix, the values of this kernel are already known, the bias are all 0 for simplicity, and the potential rough features is also known. Assuming that this is the first convolution layer, the inequality groups can be formulated as follows:

\[
\begin{align*}
x_1 + x_3 + x_7 + x_{11} + x_{13} &= 5 \\
x_2 + x_4 + x_8 + x_{12} + x_{14} &= 5 \\
x_3 + x_5 + x_9 + x_{13} + x_{15} &= 4 \\
0 &\leq x_i \leq 1
\end{align*}
\]

The solutions of these constraints are generally not unique, one of which is:

\[
\begin{align*}
x_1 &= 1, x_2 = 1, x_3 = 1, x_4 = 1, x_5 = 1 \\
x_6 &= 0, x_7 = 1, x_8 = 1, x_9 = 1, x_{10} = 0 \\
x_{11} &= 1, x_{12} = 1, x_{13} = 1, x_{14} = 1, x_{15} = 0
\end{align*}
\]

In practice, it is often the case that the constraints admit multiple solutions. If the number of the solutions is too large, our algorithm will further to solve an optimization problem (linear programming), taking the maximum or minimum value of the sum of all variables \( x_i \) as the objective function. The optimum can be directly selected as a solution.

There are various mature algorithms and tools for solving linear programs. The tool PuLP [21] which is based on simplex algorithm is employed in our experiments. If the solution fails, it means that there are obvious conflicts between the constraints, and the deconvolution algorithm can not be continued. In this circumstance, the corresponding “intermediate adversarial examples” is considered as spurious and will be discarded. If the solution can be found, it means that the decalculation of the current convolutional layer is successful. If the previous layer is another pooling layer, the Unpooling algorithm is used to process again. Otherwise, a counter-example image might be generated.

IV. EXPERIMENTAL RESULTS

A. Conv-Reluplex experiments

This experiment includes two steps. The first step is to train a CNN model. The second step is to verify the CNN model by using the Conv-Reluplex framework and generate adversarial examples, which demonstrates the effectiveness of our Conv-Reluplex algorithm.

1) Training a CNN model: The public MNIST data set [22] is employed for this experiment. MNIST is an image data set of handwritten numbers 0 to 9, including 60,000 training samples and 10,000 test samples. Each sample is a grayscale image with the size of 28×28 and the grayscale range of 0 to 255. In preprocessing, the grayscale is normalized as data between 0 and 1. The structure of the CNN model we built is shown in Fig. 6 with batch size 64 and epoch 10. After the training step, the test loss is 0.1894 and the test accuracy is 94.98%.

2) Verifying and generating picture adversarial examples by Conv-Reluplex: An image sample of handwritten numeral 6 is used to verify adversarial robustness. We input the image sample into the CNN model, extract feature data at the full connection layer, and set perturbation neighborhood \( \delta \) from 0.2 to 0.6. (Note: the perturbation neighborhood of 0.2 means that each generated value is between the original value adding 0.2 and subtracting 0.2. For example, the original input is [1.0, 1.5], and the perturbation [0.8, 1.3], [1.2, 1.7] both satisfy the setting of the perturbation neighborhood of 0.2, but [1.0, 1.8] is under the 0.3 perturbation neighborhood rather than 0.2. The verification results are shown in Table. I:

The symbol “√” indicates that there is no adversarial example here. “\” indicates that the classification is the real class of the sample, and verification can be skipped. “Fail” means that there is at least one adversarial example misclassified as the corresponding class in the specified perturbation neighborhood. Only when there is no “Fail” in an entire row, the adversarial robustness in the corresponding neighborhood is satisfied.
Table I: Verification results of a test sample of number 6 in different perturbation neighborhoods

<table>
<thead>
<tr>
<th>Perturbation neighborhood</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\delta = 0.2)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>|</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>(\delta = 0.3)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Fail |</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>(\delta = 0.4)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Fail |</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>(\delta = 0.5)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Fail |</td>
<td>√</td>
<td>Fail</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>(\delta = 0.6)</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Fail |</td>
<td>√</td>
<td>Fail</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
</tbody>
</table>

As shown in Table I, when \(\delta = 0.2\), the adversarial robustness is satisfied, and no other classification will appear. When \(\delta = 0.3\), the adversarial robustness is not satisfied, and the sample of number 6 will be classified as number 5. As the neighborhood \(\delta\) increases, the misclassification caused by perturbation becomes more and more serious. Such as \(\delta = 0.5\) or \(\delta = 0.6\), the selected sample of number 6 will be classified as numbers 5 or 8. It is obvious that the number of misclassification types and the number of intermediate adversarial examples have further increased.

An intermediate adversarial example misclassified as 5 is chosen when \(\delta = 0.6\) to carry out further analysis. The intermediate adversarial example is converted to an image by using the Unpooling and Deconvolution algorithm of Conv-Reluplex. The comparison between the original image and the adversarial example image is shown in Fig. 7. Although there is a lot of noise in the adversarial example picture, it does not prevent human from recognizing the main content.

Table II: Accuracy comparison before and after adversarial training

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy of Original Sample</th>
<th>Accuracy of Adversarial Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>94.98%</td>
<td>0%</td>
</tr>
<tr>
<td>Adversarial</td>
<td>94.89%</td>
<td>99.65%</td>
</tr>
</tbody>
</table>

In order to further analyze the difference of the model’s adversarial robustness after adversarial training, we respectively select one original sample of numbers 1, 3 and 6 as test data, and input them into the original model and the adversarial training model, then use Reluplex to verify the adversarial robustness of CNN’s full connection layer. The verification results are shown in Table III-V.

Table III: Verification results of number 1’s test sample \((\delta = 0.6)\)

<table>
<thead>
<tr>
<th>Classification</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>√</td>
<td>|</td>
<td>√</td>
<td>Fail</td>
<td>Fail</td>
<td>√</td>
<td>Fail</td>
<td>Fail</td>
<td>Fail</td>
<td></td>
</tr>
<tr>
<td>Adversarial</td>
<td>√</td>
<td>|</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>Fail</td>
<td>√</td>
<td>Fail</td>
<td>Fail</td>
<td></td>
</tr>
</tbody>
</table>
In Table III, when perturbation neighborhood $δ = 0.6$, the selected sample of number 1 has five kinds of adversarial examples in original model, the selected sample may be misclassified as 3, 4, 7, 8 or 9. However, in the adversarial training model, the selected sample only has two kinds of adversarial examples. It may be misclassified as 7 or 9. We can find that the types of adversarial examples have decreased.

In Table IV, the selected sample of number 3 has two kinds of adversarial examples in original model, which may be misclassified as 5 or 8. But in the adversarial training model, the two missclassification cases have disappeared, the selected sample has no adversarial examples in the same neighborhood ($δ = 0.6$), and the adversarial robustness has changed from not satisfied to satisfied.

In Table V, the verification situation of number 6’s selected sample is the same as number 3. It shows that this phenomenon is not accidental. Adversarial training can indeed repair the weakness of the model and enhance the adversarial robustness.

V. CONCLUSION

In this paper, to make Reluplex more practical, we propose a Conv-Reluplex verification framework, which is utilized to check adversarial robustness of CNNs. In case the robustness property is not satisfied, it generates adversarial example. Using these adversarial examples to proceed adversarial training, can indeed enhance the adversarial robustness of our model.

There are still some further work needed to be done. The complexity of the general SMT solver algorithm is exponential. Reluplex proposed its own optimization algorithm to solve some performance bottlenecks, but the efficiency is severe limited by the high nonlinearity of the resulting formulas. So we can only deal with some smaller networks at present. The network trained by MNIST only has more than 4,000 relu nodes, while the network nodes trained by ImageNet are too many to handle. We hope to do some work to improve efficiency so that it can handle more network nodes, and it is one of our goals to handle color pictures. Finally, the recurrent neural networks (RNNs) can be taken into the consideration in both original Reluplex and the extensions of Reluplex.

ACKNOWLEDGMENT

We acknowledge the support of National Natural Science Foundation of China (NSFC) Project 61972284.

REFERENCES

ConvolutionalNeuralNetwork.
[21] https://pythonhosted.org/PuLP/
Tree Ensemble Property Verification from A Testing Perspective

Bohao Wang¹,², Zhe Hou³, Gelin Zhang², Jianqi Shi², and Yanhong Huang* ¹,²

¹Shanghai Key Laboratory of Trustworthy Computing, Shanghai, China
²National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai, China,
³Griffith University, Australia, z.hou@griffith.edu.au

Abstract—With the development of artificial intelligence, machine learning algorithms are currently being used in more and more fields, such as autonomous driving, medical diagnosis, etc. In recent years, much research focuses on property verification of machine learning models. As one of the machine learning models, the tree ensemble model's structure is amicable to formal verification, but large models still prove hard to verify due to the combinatorial path explosion. This paper presents a violation-driven, sound but incomplete method from a testing perspective. We generate an explanation model of the original model and verify it formally. After a narrowed search space is obtained, we verify the original model by a testing-based method. A counterexample is then proof that the original model violates the property. We elaborate our method through a case study in detail.

Index Terms—Tree Ensemble, Property Verification, Testing

I. INTRODUCTION

Nowadays, artificial intelligence utilizing machine learning algorithms has achieved a lot of success in many fields, such as face recognition, autonomous driving, medical diagnostics, etc. With the application of these technologies in more and more fields, people doubt whether they can meet certain properties, such as security, robustness, fairness etc., since deep learning models are almost a black-box, and the tree ensemble models are also too complex. In recent years, deep learning has developed rapidly, and now there has been much research on the verification of neural network [1], [2]. This paper is mainly focused on whether the tree ensemble models satisfy specific interesting properties.

Suppose you have a random forest model that tells you whether or not two cars will collide based on a variety of features. Traffic safety experts suggest that when the distance between two cars is less than 5 meters and both cars' speed is greater than 120km/h, the two cars are bound to collide. The prediction of a random forest model may be correct 99% of the time, but it may violate this property in some input spaces, so we still need to verify the random forest model.

As background work, we tried to develop and implement a general, sound and complete verification algorithm for random forest. Our method divided the input space of the random forest into different disjoint sets. Suppose the random forest has 10 trees, each tree has 32 leaf nodes (depth is 5), and each leaf node corresponds to a branch. We join each tree branch with one branch of all other trees to represent the disjoint sets of the input space divided by the random forest model. When the input space is divided into disjoint sets, the specific region of input space that does not satisfy the property will be found by a property checking algorithm. The method mentioned above faces combinatorial path explosions, which means it is not scalable and can only handle random forests where the sum of the number and depth are no more 15 in our experiment. Random forest models on this scale are toys that do not work in real life. Similar results are obtained in recent work on verifying tree ensembles [3].

Consequently, we take a step back and sacrifice completeness for a more feasible approach. The proposed approach in this paper performs verification of tree ensembles from a testing perspective. We first transform the original tree ensemble model into a relatively simple model, which is referred to as an “explanation model”. The explanation model has a highly similar predictive behaviour compared to the original model [4]. Then we verify the explanation model against a property. Our hypothesis is that when the explanation model violates the property in a certain search space, it is highly likely that the original model also violates the property in the same search space. Then we can narrow down the search space using the explanation model and verify the original model in a much smaller input space by a test-based method. If a counterexample is found, then it is proved that the original model violates the property.

The contributions of this paper include:

• We propose a method for tree ensemble property verification, which alleviates combinatorial path explosion.
• We have developed a tool called TEPV that can perform property checking on tree ensemble models.

*Corresponding Author
DOI reference number: 10.18293/SEKE2021-087
• We validate our approach on datasets of different sizes, and the experiment demonstrates that our approach is scalable and works well on large tree ensemble models.

The rest of this paper is organized as follows: Section II gives the background knowledge of this paper. Section III details the proposed method. Section IV demonstrates a case study and experiment of our approach. We discuss related work in Section V. Finally, we conclude and give some future work in section VI.

II. PRELIMINARIES

This section presents the required background knowledge, including decision trees, ensemble of decision trees, and the explanation tree ensemble model. We also give the definition of the properties used in this paper.

A. Decision Trees With a Logical Foundation

We adopt the definitions of decision tree described in [4]. In supervised learning, a structured dataset for classification is defined as set of instances of the form \((\vec{x}, y)\) where \(\vec{x} = [x_1, ..., x_n]\), \(n \in \mathbb{N}\), is an input vector called features and \(y\) is an outcome value often called the label. We denote by \(X\) the feature space and \(Y\) the outcome space.

![Decision Tree Diagram](image)

A decision tree is composed of internal nodes (diamonds in Figure 1) and terminal nodes called leaves (ovals in Figure 1). Each internal node is associated with a logical formula over a feature. Each leaf node contains a set of instances, which yield a feature. Each leaf node contains a set of instances, which yield a feature.

Each internal node is associated with a logical formula over the properties used in this paper. Each leaf node contains a set of instances, which yield a feature.

Given a decision tree, any input vector (or instance) is associated with a single leaf. A decision tree is, therefore, a compact representation of a function of the form \(t : X \rightarrow \mathbb{N}^m\), where \(m\) is the number of classes. The output of a decision tree is a distribution of votes for each class. To obtain an outcome in \(Y\), we take the class with the most votes.

B. Random Forest

We adopt the definitions of Cui et al. [5]. Let an ensemble be a set of decision trees of size \(T\). It gives the weighted sum of the trees as follows:

\[
E(x) = \sum_{i=1}^{T} w_i \cdot t_i(x)
\]

where \(E\) is the function for the ensemble, \(w_i\) and \(t_i\) are respectively the weight and function for each tree. The summation aggregates the weighted votes from each tree and obtains the final votes for each class. Thus, the ensemble is also a function of the signature \(E : X \rightarrow \mathbb{N}^m\) and requires a voting mechanism to obtain the outcome. We mainly focus on the ensemble trees by bagging. Each decision tree is trained using a subset of the dataset that is sampled uniformly with replacement. The remaining instances form the out-of-bag (OOB) set. When selecting the best formula at each decision node in a tree, only a subset of the features are considered. This is commonly found in algorithms such as Random Forest [6]. Bagging grows large trees with low bias, and the ensemble reduces variance.

C. Explanation Tree Ensemble

In our previous work [4], we extract logical formulas from an original tree ensemble model \(\mathcal{M}_o\) to synthesize an explanation model \(\mathcal{M}_e\). \(\mathcal{M}_e\) is an approximation of \(\mathcal{M}_o\), and it contains a set of decision rules. A decision rule is a tuple \((F, s, w)\), where \(F\) is a classical logic formula, \(s\) is the signature, which is a normalized vote distribution between classes, and \(w\) is the weight, which indicates the importance of the rule. There are four parameters that determine the generation of the \(\mathcal{M}_e\), which are \(\theta\), \(\phi\), \(\psi\) and \(k\). \(\theta\) determines the complexity of the decision rules, \(\psi\) determines the signature \(s\) of each decision rule. Both \(\phi\) and \(k\) determine the number of the decision rule in \(\mathcal{M}_e\). Refer to [4] for the details.

The explanation model \(\mathcal{M}_e\) can make predictions based on the following method: given a data instance \(x\), find all the decision rules in \(\mathcal{M}_e\) whose logical formula \(F\) is satisfied by \(x\), then multiply the signature of those rules by their corresponding weight, and add them up to get a tuple of vote distributions. The class with the largest value is the output. The above procedure is denoted as \(\mathcal{M}_e(x) = c\), where \(c\) is a class. Figure 2 gives an example of how \(\mathcal{M}_e\) makes a prediction.

D. Properties of Interest

In this paper, we consider a general class of properties, which is defined below.

**Definition 1:** (Properties) Let \(f : X^n \rightarrow \mathbb{R}^m\) be the function to be verified. A property is of the form \(P : Constraint \rightarrow \mathbb{R}\).
There are three stages in our method. First, we generate an explanation model. The overview of our method is given in Figure 3. Here, fidelity refers to the degree of similarity between the predictions of $e$ and $o$ on unseen data [7]. The second stage is the ExModel Checker part. In this step, our approach checks whether the property $P$ is violated by $M_e$. If the ExModel Checker finds that $P$ is violated, the process will continue. Finally, OrModel Checker will verify $M_o$ against the property $P$. If $M_o$ violates $P$, we will output a decision rule explanation and a counterexample. The counterexample is an instance that $P$ is not satisfied by $M_o$, and the decision rule explanation is a narrowed-down search space.

III. METHODOLOGY

This section details our tree ensemble property checking approach. The overview of our method is given in Figure 3. There are three stages in our method. First, we generate an appropriate explanation model $M_e$. Here, fidelity refers to the degree of similarity between the predictions of $M_e$ and $M_o$ on unseen data [7]. The second stage is the ExModel Checker part. In this step, our approach checks whether the property $P$ is violated by $M_e$. If the ExModel Checker finds that $P$ is violated, the process will continue. Finally, OrModel Checker will verify $M_o$ against the property $P$. If $M_o$ violates $P$, we will output a decision rule explanation and a counterexample. The counterexample is an instance that $P$ is not satisfied by $M_o$, and the decision rule explanation is a narrowed-down search space.

### A. STAGE 1: Generate Explanation Model

We adopt our previous work [4] to generate the explanation model $M_e$. The process of generating explanation model is expressed as the following formula:

$$
\text{explain}(M_e, \theta, \phi, \psi, k) = M_e
$$

A major difference compared to the previous work is that the $M_e$ we generate here does not need to follow the criteria of the explanation should be concise and small. Instead, we only follow one criterion: the classification behavior of the explanation model $M_e$ should be as similar as possible to the original model $M_o$.

We mentioned in Section II-C that different parameters generate different $M_e$. In order to get an explanation model $M_e$ with high fidelity, we also use the linearly decreasing inertia weight particle swarm optimization algorithm (LDIWPSO) [8] to optimise the four parameters: $\theta, \phi, \psi$ and $k$. Since our criteria are different from [4], we propose a new equation below as the fitness, i.e., the objective function to be optimised.

$$
S_{opt} = \sum_{i} \Theta(M_e(x_i), M_o(x_i))
$$

where $\Theta(x, y)$ is a function that outputs 1 when $x = y$; otherwise outputs 0, $x_i$ is the sample in the test set, and $n$ is the number of samples in the test set. After optimization, we finally obtain $M_e^{opt}$.

### B. STAGE 2: Verify Explanation Model

Once we obtain $M_e^{opt}$, we can use the ExModel Checker to verify $M_e^{opt}$ against properties. The detail of Stage 2 is shown in Algorithm 1.

#### Algorithm 1 Stage 2: find violation of explanation model

1. **Input:** Property $P$, Explanation Model $M_e^{opt}$
2. **Output:** a set of decision rules set $\Phi$
3. $R \leftarrow \emptyset$
4. $\Phi \leftarrow \emptyset$
5. for all $r \in M_e^{opt}$ do
6. if $r \land P \triangleq \text{constraint is satisfiable then}$
7. $R \leftarrow R \cup \{r\}$
8. end if
9. end for
10. for all $R' \subseteq R$ do
11. if $R'$ is satisfiable and $M_e^{opt}.\text{predict}(R') \neq P.\text{target}$ then
12. $\Phi \leftarrow \Phi \cup \{R'\}$
13. end if
14. end for
15. return $\Phi$

Given an $M_e^{opt}$ and a property $P$, the first step of checking $M_e^{opt}$ is rule selection, corresponding to the lines 4-8 in Algorithm 1. For simplicity, the notation of $M_e^{opt}$ not only refers to the explanation model but also the decision rules in the explanation model. For the decision rules in $M_e^{opt}$ that
are compatible with the constraint of property P, we collect them as R. After rule selection, we try to find the subset of R whose decision rules are satisfying when combined. Assume there is a set R′ that meets the requirements; we use M′opt to make a prediction based on R′. We mentioned in Section II-C that the explanation model would first select decision rules in M′opt are satisfied by x, and then make a specific prediction according to the selected decision rules. Here, the prediction of M′opt omits the steps for selecting decision rules, as these are done in lines 4 - 8 already. Instead, we directly give the prediction based on the set of R′.

If the prediction result is not consistent with the target of the property P, M′opt violates the property P. Then, we will continue to verify the original model M_o using OrModel Checker based on the narrowed search space.

C. STAGE 3: Verify Original Model

Because M′opt and M_o have very similar predictive behavior, when we look at the same (narrowed-down) search space, if M′opt violates the property P, it is very likely that M_o may also violate P. We present the details of stage 3 in Algorithm 2.

Algorithm 2 Stage 3: find violation sample of original model.

1: Input: Property P, Original Model M_o, decision rules set R′ and the number of generated samples n
2: Output: flag, c and e
3: flag ← False
4: c ← None
5: e ← R′ ∧ P.constraint
6: samples ← generate n samples based on e
7: for all s ∈ samples do
8: if M_o.predict(s) ̸= P.target then
9: flag ← True
10: c ← s
11: return flag, c, e
12: end if
13: end for
14: return False, None, None

Note that the constraint component of a property has the same structure as the logical formula of a decision rule, so they can be used in conjunction to limit the search space. More specifically, we combine the decision rules in R′ and the property P’s constraint and simplify them by merging subFORMElae on the same feature (Line 5 of Algorithm 2). At this point, the counterexample search space of property P is reduced significantly by the above simplified formula, which we call decision rule explanation. Then, a sample generator will generate many samples in the narrow search space.

We adopt a Generative Adversarial Network (GAN) [9] model as the sample generator. GAN has two parts: a discriminative network (the discriminator) and a generative network (the generator). The discriminator learns to distinguish the features of a given data instance. The generator, which learns to confuse the discriminator, can generate fake but plausible data via an adversarial learning process. GAN will generate samples with the same distribution as the training set, but we need samples within the narrow search space determined by e. So we add a filter after the GAN model to select samples that meet the requirements. The process mentioned above corresponds to Line 6 of Algorithm 2.

Next, we use M_o to make predictions from the generated samples, which are expected to have a high chance of violating P. If the predicted result is indeed different from the target of the property, it means that M_o violates the property. The sample with the different predicted result is provided as the counterexample, and the final narrow search space is output as a decision rule explanation.

Noted that there may be a lot of R′ that cause property violation of M′opt in the subset of R, which means a lot of decision rules sets will be emitted in stage 2, but what we want to find is a R′′ that cause property violation of both two models. Since our approach is violation-driven, once we find one appropriate R′′, our method will stop and return the counterexample and decision rule explanation. Otherwise, the method will continue to find the violation.

IV. EXPERIMENT AND CASE STUDY

We implement our method in Python and develop a tool called TEPV (Tree Ensemble Property Verification). We use scikit-learn [10] to train random forest models and evaluate our tool TEPV through a case study and experiment. For random forest in our experiment, the number of trees is 100, and the depth of the tree is unlimited. For simplicity, we only consider binary classification datasets with numeric features. However, the proposed approach can easily be extended to multi-class datasets with both numeric and nominal features. We test our method on three datasets: diabetes, JM1 and MiniBooNE, all of which are available on OpenML [11]. Particle size and iteration period are set to 20 by default, which are two parameters in LDWW-PSO. Experiments were conducted on a machine with an Intel Core i9-7960X CPU and 32GB RAM.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Constraint of the Property</th>
<th>Rule Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>preg</td>
<td>lower bound: 7.51</td>
<td>upper bound: 10.65</td>
</tr>
<tr>
<td>plas</td>
<td>138.52</td>
<td>+∞</td>
</tr>
<tr>
<td>pres</td>
<td>99.02</td>
<td>101.23</td>
</tr>
<tr>
<td>insu</td>
<td>77.94</td>
<td>93.51</td>
</tr>
<tr>
<td>skin</td>
<td>26.94</td>
<td>60.39</td>
</tr>
<tr>
<td>age</td>
<td>0.63</td>
<td>2.23</td>
</tr>
</tbody>
</table>

A. Case Study 1: Pima Indians Diabetes Dataset

The Pima Indians Diabetes dataset [12] has 768 samples, 8 features, 2 classes. The features are as follows: the number of times pregnant (preg), plasma glucose concentration (plas), diastolic blood pressure (pres), 2hour serum insulin (insu), triceps skinfold thickness (skin), body mass index (mass), diabetes pedigree function (pedi) and age. Two classes are
positive and negative. We use 668 samples to train a random forest and 100 samples as the testing set.

In this case study, a doctor wants to check if the random forest model \( M_o \) satisfies his expert knowledge. He puts forward a property as follows. When a patient meets the constraint that preg is between 7.51 and 10.65, plas is bigger than 138.52, pres is between 99.02 and 101.23, insu is between 77.94 and 93.51, skin is less than 99.04, mass is between 26.94 and 60.39, and pedi is between 0.63 and 2.23, she should be diagnosed as positive. The doctor argues that the machine learning model is unreliable if it violates this property which is derived from his expertise.

We use our tool TEPV to verify the random forest model \( M_o \) against the property put forward by the doctor. We generate explanation model \( M_e^{opt} \) as Table II shows. And the decision rules in red is one set of the output of Algorithm 1. Based on Algorithm 1 and Algorithm 2, we have narrowed the search space and provide a counterexample of violation. In Table I, compared to the constraint of the tested property \( P \), it is clear that the upper bound of preg, mass, pedi and age shrinks; in the meantime, the lower bound of plas and mass rises.

### TABLE III

A COUNTER EXAMPLE OF THE TESTED PROPERTY

<table>
<thead>
<tr>
<th>preg</th>
<th>plas</th>
<th>pres</th>
<th>insu</th>
<th>skin</th>
<th>mass</th>
<th>pedi</th>
<th>age</th>
<th>predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>157.64</td>
<td>99.99</td>
<td>86.62</td>
<td>47.19</td>
<td>28.61</td>
<td>0.66</td>
<td>21</td>
<td>negative</td>
</tr>
</tbody>
</table>

The counterexample that reflects the violation of \( M_o \) is shown in Table III. This sample is generated by our sample generator in the input space defined by the decision rule explanation. Our method narrows down the search space, and the counterexample found in this space reflects the effectiveness of our approach.

### B. Experiment

Under normal circumstances, users want to test whether the model violates certain user-specified properties that may be domain-specified. Since we are not domain experts and can not define properties with professional knowledge, we use the following method to randomly generate properties.

Assume the dataset has \( n \) features and \( m \) classes, we first find the \( \min_i \) and \( \max_i \) of feature \( f_i \) \( i \in \{1, \ldots, n\} \), which is the minimum and maximum of the \( f_i \) in the training samples. For the constraint of the property, we select \( f_i \) with a chance of \( p_1 \). And for each \( f_i \) selected, randomly generate \( \alpha_i \) and \( \beta_i \) between \( \min_i \) and \( \max_i \), assume \( \alpha_i < \beta_i \). Then we generate the constraint \( \alpha_i \leq f_i \leq \beta_i \) with a chance of \( p_2 \% \), otherwise generate the constraint \( \alpha_i \leq f_i \) or \( f_i \leq \beta_i \) with 50% chance each. For the target of the property, we randomly select one class as the target with equal probability.

We test on three datasets: diabetes, JM1 and MiniBooNE. JM1 is a dataset about software defect prediction; it has 21 features and 10k samples. MiniBooNE dataset is used to distinguish electron neutrinos (signal) from muon neutrinos; it has 50 features and 130k samples. For each dataset, we train a random forest model with 100 trees of unlimited depth, which are the usual default settings in real-life applications. We set \( p_1 \in \{0.5, 0.8\} \), \( p_2 \in \{0.5, 0.8\} \) and randomly generate 25 properties of every \((p_1, p_2)\) combination, totalling 100 properties for each dataset. The number of samples generated by the sample generator is 1000. For JM1 and MiniBooNE, 90% of the samples are used as the training set, and 10% of the samples are used as the testing set. For each property, if no counterexample is found or if the process takes more than one hour, we consider it a failure and the property is skipped, and the program begins to verify the next property.

### TABLE IV

PROPERTIES CHECKING ON DIFFERENT DATASETS.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>( M_e^{opt} ) Fidelity Parameters ((\theta, \phi, \psi, k))</th>
<th>Acc.</th>
<th>Vio.</th>
<th>Time</th>
<th>Std.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diabetes</td>
<td>( 93% ) ((0.57, 0.1, 0.67, 12.0) )</td>
<td>80%</td>
<td>75%</td>
<td>388</td>
<td>1424</td>
</tr>
<tr>
<td>JM1</td>
<td>( 92% ) ((0.66, 0.14, 0.7, 12.0) )</td>
<td>92%</td>
<td>60%</td>
<td>354</td>
<td>600</td>
</tr>
<tr>
<td>MiniBooNE</td>
<td>( 88% ) ((0.8, 0.04, 0.03, 20.0) )</td>
<td>93%</td>
<td>58%</td>
<td>524</td>
<td>752</td>
</tr>
</tbody>
</table>

The results of the experiment are shown in Table IV. \( M_e^{opt} \) Fidelity is the proportion that \( M_e^{opt} \) and \( M_o \) predict.
the same results in the test set. Acc. is the prediction accuracy of $M_o$ on the test set. Vio. is the percentage of the properties for which we can find a counterexample within timeout. Time is the average time of finding counterexamples (in seconds), and Std. is the standard deviation of the time. The results show that our tool, TFPV, can find some violation of tree ensemble in different datasets. When the model violates properties, the average time required is less than 20 minutes. Since the properties are generated randomly, a large standard deviation is acceptable. But it is clear that our approach is scalable and works well on datasets of different sizes and large tree ensemble models.

V. RELATED WORK

Since machine learning algorithms are used more and more frequently in daily life, people have some doubts about whether they meet certain requirements or properties. Therefore, more and more researchers are now working on the property verification of machine learning models. In the following, we will introduce the property verification of machine learning models in two parts: verification of deep learning models and verification of tree ensemble models.

A. Verification of Deep Learning Models

In 2010, Pulina et al. [2] proposes abstract interpretation to verify a DNN and introduces a linear approximation algorithm to estimate the interval of ReLU and Sigmoid output. Fichetti et al. [13] propose a 0-1 MILP encoding to model a DNN for property verification and reasons through a MILP solver and implements a bound tightening mechanism to reduce the search space.

A method for verification of feed-forward neural networks with piecewise linear activation function was presented in [14]. They treat the neural network model as a block-box and use the SMT solver to verify the approximation of the block-box. Compared with Ehlers, Huang et al. [1] describe a white-box approach to verify the feed-forward neural networks and introduce a feed-forward analysis that partially based on discretization to test robustness and find adversarial examples.

B. Verification of Tree Ensemble Models

Tree ensemble models are non-continuous step functions, which is different from neural networks in deep learning. Therefore, the techniques mentioned above cannot be used to verification of tree ensemble models. Recently several researchers have pursued approaches to the verification of tree ensemble models.

Tornblom et al. [3] proposed a robustness verification tool of tree ensembles called VoTE. Their method is an abstraction-refinement procedure that iteratively refines a partition of the input space where each block of the partition is a hyperrectangle. A tool named Silva introduced by Ranzato and Zanella [15] pushes forward the line of research by designing a general and principled abstract interpretation-based framework for the formal verification of robustness and stability properties of decision tree ensemble models. Unlike tool VoTE, the soundness and completeness properties of Tornblom’s verification algorithm are not formally proved. The algorithm of Silva is based on the principles of abstract interpretation, which is endowed with a formal soundness and completeness proof.

VI. CONCLUSION AND FUTURE WORK

This paper presents a method for verifying whether a tree ensemble model violates a user-specific property. We give one case study and show that our approach works. Moreover, in the experiment, we test various properties on datasets of different scale, which reflects the effectiveness of our method. In addition, the number of trees in the tree ensemble we tested is 100, and the depth is unlimited. By contrast, related methods struggle to verify 25 trees of depth 20 [3]. This demonstrates that our approach is scalable. Our approach relies heavily on sample generation, and in some certain cases, the performance of our tool, TEPV, may degrade dramatically. As future work, we plan to improve the sample generation algorithm so that our method can produce samples that meet the requirements more quickly and improve the speed of finding counterexamples.

REFERENCES

DeepAuto: A First Step Towards Formal Verification of Deep Learning Systems

Yuteng Lu*, Weidi Sun*, Guangdong Bai† and Meng Sun*

*School of Mathematical Sciences, Peking University, Beijing, China
Email: {luyuteng, weidisun, sunm}@pku.edu.cn
†School of Information Technology and Electrical Engineering, The University of Queensland, Brisbane, Australia
Email: g.bai@uq.edu.cn

Abstract—Deep Learning (DL) offers a data-driven programming paradigm in which Deep Neural Networks (DNNs) can be constructed through a set of training data. It has been widely adopted in many real-world applications. However, many studies have shown that DL systems suffer from adversarial attacks, especially when they are applied to security- and safety-critical domains. Given that formal verification has proved a great success in many areas such as software engineering, using it to achieve a high-level security assurance in DL systems is considered promising. In this paper, we design and implement DeepAuto which makes the significant bridge between automata and DNNs. With the aid of DeepAuto, we demonstrate how DNNs can be modeled as automata and be verified formally in the widely used model checker UPPAAL. The potential usefulness of DeepAuto shows the connection between DNNs and automata and provides a solution for the construction of more trustworthy DL systems.

Index Terms—Formal Verification, Deep Neural Network, Timed Automata, DeepAuto

I. INTRODUCTION

Deep Learning (DL) has enjoyed tremendous success over the past few years, achieving or exceeding human-level performance in various areas, including security-critical applications, such as autonomous vehicles [3], computer vision [6], speech recognition [10], robotics and competitive games such as Go [11]. However, DL systems often exhibit incorrect and unexpected behavior [12], carrying the risk of endangering human lives such as a fatal accident of self-driving car [9]. Thus, more concerns have been raised about the wide adoption of DL systems in security- and safety-critical systems.

To mitigate such concerns, one of the most active research areas in recent years is to design testing coverage for DL systems. Nevertheless, testing could be a quality metric for DL systems, but it cannot guarantee the correctness of the systems, which means numerous blunders may be concealed in DL systems and the testing approach cannot prove the absence of such errors. Towards addressing the aforementioned limitations, the application of formal verification techniques in DL systems appears to be a potential solution. In those traditional areas like software engineering, formal verification has achieved great success in guaranteeing that a system is free of certain defects or satisfies certain properties, and has played an important role in ensuring the correctness and reliability of increasingly complex software and hardware systems [4].

In the DL area, various types of DNNs have been widely used, such as Feed-forward Neural Networks (FNNs), Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Among these networks, a problem existing ubiquitously is that the output result may deviate from the expectation because of small perturbations to the input, which are so-called adversarial examples [7]. We need to ensure that DNNs deployed in security- and safety-critical domains satisfy expected properties against such infinitely many perturbations. In this work, we propose a formal framework named DeepAuto. It extracts the data flow of DNNs and name the parameters in DNNs, which results in expressible properties. It then models the data flow in automata, and converts the properties of DNNs into the properties of automata.

Since timed automata is a special type of automata and modeling RNNs needs to consider the temporal properties, we use timed automata for modeling DNNs in this paper. We link timed automata and DNNs, model the data flow process inside DNNs with timed automata. With the help of timed automata, we can observe the behavior inside the DNNs step-by-step and verify the properties of DNN-based systems. The model checker UPPAAL [2] is a tool designed to verify systems that can be modeled as timed automata. In UPPAAL, we can assign arbitrary value to input relying on demand. We use UPPAAL in our work to put DeepAuto into practice. Overall, our contributions are summarized as follows:

1) We introduce DeepAuto, the first white-box modeling and formal verification framework for DL systems. DeepAuto is practical and universal, and could be used to model trained DNNs. (2) DeepAuto provides a methodology to reduce the complex problem of formally verifying DNNs into a model checking problem which can be addressed by off-the-shelf model checkers such as UPPAAL. (3) Our work reveals the connection between DNNs (i.e., FNNs, CNNs and RNNs) and timed automata, and sheds light on applying formal verification to DL systems.

II. DATA FLOW AND TIMED AUTOMATA

To systematically ensure the correctness and reliability of DNNs in safety- and security-critical domains, it is crucial to apply formal verification techniques on DNNs. We first give the formal definition of DNNs as follows.
Definition 1 (Deep Neural Network): A Deep Neural Network is a tuple \( N = (L, W, F, T) \) where \( L = \{ l_1, l_2, \ldots, l_n \} \) is a sequence of layers and each layer \( l_k \) contains \( s_k \) neurons, \( W = \{ w_1, w_2, \ldots, w_{n-1} \} \) is a sequence of matrices and the elements of matrices are weights between layers, \( F = \{ f_2, f_3, \ldots, f_n \} \) is a sequence of activation functions, and \( T \) is the time variable which is needed in RNN.

Intuitively, the properties of DNNs can be affected by \( W \) and \( F \). To specify and verify properties of DNNs formally, we define the data flow, which is a record of variable changes within a DNN. Let \( X = \{ X_1, X_2, \ldots \} \) be a sequence of variable vectors in each stage of the feed-forward process for a given DNN \( N \), where \( X_1 \) is a set of variables in input stage and \( X_{i+1} \) is dependent on \( X_i \) within \( N \). The definition of data flow is as follows.

Definition 2 (Data Flow): For a given DNN \( N \) and each \( X_i \), there is a function \( \phi_i \) mapping \( X_i \) to \( X_{i+1} \), i.e., \( X_{i+1} = \phi_i(X_i) \). We call \( X_{i+1} = \phi_i(X_i) \) a data transformation. The sequence of all such data transformations is a data flow.

Since there are multiple mapping relationships between \( X_{i+1} \) and \( X_i \) in different DNNs, we use \( X_{i+1} = \phi_i(X_i) \) to represent this data transformation process uniformly. Data flow records the transformation process of \( X \), and we can investigate the interior of DNNs thoroughly based on data flow. We extract the data flow within DNNs and model DNNs as timed automata. Timed automata are proposed to model the behavior of systems over time, expanding clock variables on the basis of finite-state automaton. It is decidable to verify whether states in timed automata are reachable.

Definition 3 (Timed Automaton): A timed automaton is a tuple \((S, s_0, A, C, G, E)\), where \( S \) is a finite set of states, \( s_0 \subseteq S \) is a set of initial states, \( A \) is a finite set of actions, \( C \) is a finite set of clocks, \( I : S \rightarrow \Phi(C) \) assigns invariant to each state, and \( E \subseteq S \times A \times 2^C \times \Phi(C) \times S \) is a set of transitions between states.

A switch \((s, a, \lambda, \varphi, s')\) represents an edge from state \( s \) to state \( s' \) with an action \( a \), where \( \varphi \) is a clock constraint over \( C \) that specifies when the switch is enabled, and \( \lambda \subseteq C \) gives the clocks to be reset with this transition.

Through the above definitions, we can easily derive the mapping relationships between DNNs and time automata. The update processes within DNNs can be mapped to update assignments (i.e., actions) in automata’s edges, and the layers of DNN can be mapped to states of automaton.

III. OVERVIEW OF DEEPAUTO

An overview of the DeepAuto framework for modeling and verifying DNNs is shown in Fig. 1. We first use weights and activation function of a given DNN to construct the data flow. Then, the DNN is modeled by timed automaton, and its properties are formalized and converted into the properties of timed automaton. Finally, we can check whether the DNN meets expected properties. The pseudo-code for DeepAuto is given as in Algorithm 1.

Algorithm 1 takes the weights of DNN (the \( W \)), activation function of DNN (the \( F \)), structure information of DNN (e.g., the number of layers in \( N \)), and expected property as inputs. In addition, circulation of time round is also needed for input when modeling RNN. Lines 1 to 6 of Algorithm 1 describe the abstract and model process, whose details are given later in Section IV. After obtaining the automaton, we can convert the properties and construct queries needed for the model checker, and finally use the model checker to query whether DNN meets the expected properties. If so, the DNN must satisfy the verified properties. Otherwise, users can analyze and debug the DNN according to unsatisfied properties.

IV. MODELING OF DNN

More details of the modeling process are given in this section. Since commonly used DNNs (i.e., FNN, CNN, and RNN) have different internal structures, there are differences in the specific modeling processes.

A. Modeling of FNN

We use a FNN with a hidden layer as an example. The hidden layer has two neurons. The weights of the neuron in the input layer to the two neurons in the hidden layer are 1, -1 respectively. The rest of the weights are all 1. The activation function is ReLU function. When the input value is non-negative, the output value is always identical to the input value.

The given FNN can be modeled as automaton based on Algorithm 1. In all the algorithms of this work, the index of layer \( l_{i+1} \) is 1. Using weights, structure information and activation
Algorithm 2 Modeling FNN (CNN) as automaton

Input: The weights of FNN (CNN) \( W \); List of layers for FNN (CNN) \( L = \{l_1, l_2, \ldots, l_n\} \); Activation functions of FNN (CNN) \( F = \{f_2, f_3, \ldots, f_n\} \)

Output: Abstract automaton \( A_{\text{NN}} \)

1. \( node_1 \leftarrow L[0] \)
2. for index \( i \) of layers do
   3. if \( L[i] \) is Convolution layer then
      4. \( Update \leftarrow X_{i+2} = W_i \times X_{i+1} \)
      5. end if
   6. if \( L[i] \) is Max-pooling layer then
      7. \( Update \leftarrow X_{i+2} = \text{Max}(X_{i+1}) \)
      8. end if
   9. if \( L[i] \) is Activation layer then
      10. \( Update \leftarrow X_{i+2} = f_i(W_i \otimes X_{i+1}) \)
   11. else
      12. \( Update \leftarrow X_{i+2} = W_i \otimes X_{i+1} \)
   13. end if
14. \( T_{(i+1)(i+2)} \leftarrow Update \) \(^{\otimes} T_{(i+1)(i+2)} \) is the transition between \( node_{i+1} \) and \( node_{i+2} \).
15. \( node_{i+2} \leftarrow L[i+1] \)
16. end for
17. \( T_{(n+1)(1)} \leftarrow \text{Guard()} \)
18. return \( A_{\text{NN}} \)

Fig. 2. FNN Model

In this example, \( node_1, node_2, \) and \( node_3 \) correspond to the states Input_layer, Hidden_layer, and Output_layer in Fig. 2 respectively. We construct equations, which are actually data flows, such as \( X_{i+2} = f(W_i \otimes X_{i+1}) \), \( X_{i+2} = W_i \otimes X_{i+1} \). Each of these equations is assigned to the Update of a transition between states that are created based on the layer information of the FNN. Specifically, if Update corresponds to hidden layer, the data flow assigned should be \( X_{i+2} = f(W_i \otimes X_{i+1}) \). Otherwise, the data flow assigned should be \( X_{i+2} = W_i \otimes X_{i+1} \).

Guard() is a function with an if-else structure, whose returned type is Boolean. We abstract the properties that DNNs need to satisfy into quantitative relationships by the aid of data flow. After generating such quantitative relationships, we encode them into the conditions of if-else statements, which enable us to transform the verification of DNNs’ properties into the verification of automata’s properties. The process of constructing Guard() is similar to constructing formal specifications in traditional software, and requires prior knowledge of the system.

B. Modeling of CNN

Algorithm 2 can also be used to model CNNs with the common structure (i.e., CNNs with convolution layers and max-pooling layers), which takes the weights of CNN, structure information and activation function as inputs and produces an automaton as output. The data flows in CNNs have forms like \( X_{i+2} = f(W_i \otimes X_{i+1}) \), \( X_{i+2} = W_i \otimes X_{i+1} \) or \( X_{i+2} = \text{Max}(X_{i+1}) \). According to the type of each layer, the data flows are assigned to the corresponding transition Updates. Specifically, the above-mentioned data flows correspond to activation layer, convolution layer and max-pooling layer respectively. The structure of automaton is constructed using structure’s information.

Guard() is also a function with an if-then-else structure and its return type is Boolean. Based on the properties of CNNs, we can obtain quantitative relationships. By encoding such relationships into the condition of the if-then-else statement, we transform the verification of CNNs’ properties into the verification of automata’s properties. We use a CNN with one convolution layer, one activation layer and one max-pooling layer as the running example. The corresponding automaton in UPPAAL is as shown in Fig. 3.

C. Modeling of RNN

Using Algorithm 3, we can model Vanilla RNNs of any size and simulate their running process, so as to check the properties. Algorithm 3 takes time round of RNN, RNN’s weights, structure information and activation functions as inputs and produces an automaton simulating RNN as output. Based on the number of time round and weights within RNN, we can construct value update statements Propagate\(_1\)(time), \ldots, and Propagate\(_n\)(time). Whereafter, time update statement time = time + 1 and Propagate\(_n\)(time) will be assigned to \( T_{12} \). Simultaneously, Propagate\(_2\)(time), \ldots, and Propagate\(_n\)(time) will be assigned to the corresponding transitions. We use \( t <= \text{time} \& \text{time} < t_o \) to control rounds of the RNN. The approach to construct Guard() is the same as that in FNN and CNN modeling. By adding Guard() to the transition, we can affect the property of automaton.

Here we use a three-cycle RNN as an example. We assume that weights of the RNN is \( \{1, 1, 1\} \) and activation function is ReLU. The time round \( t_o \) is set as 3. To model such an RNN, we follow Algorithm 3. We first construct Propagate\(_1\)(time) and Propagate\(_2\)(time) to simulate the data processing. Time update statement time = time + 1 and Propagate\(_1\)(time) are assigned to the transition from Input_layer to Hidden_layer, and Propagate\(_2\)(time) is assigned to the transition from Hidden_layer to Output_layer. Using \( t <= \text{time} \& \text{time} < t_o \), we limit the number of rounds to 3. After time 3, the transition from the state Output_layer to the state Input_layer will pass from the state Success because of the invari-
Algorithm 3 Modeling RNN as automaton

Input: Time round of RNN \( t_0 \); The weights of RNN \( W \);
List of layers for RNN \( L = \{l_1, l_2, \cdots, l_n\} \); Activation
functions of RNN \( F = \{f_2, f_3, \cdots, f_n\} \)

Output: Abstract automaton \( A_{\text{RNN}} \)

1: \( \text{node}_1 \leftarrow L[0] \)
2: \( L.\text{append}(\text{Success}) \)
3: for Time round of RNN \( t \) do
4: \( \text{Construct Propagate}_1(t), \cdots, \text{Propagate}_{n-1}(t) \)
5: end for
6: for index \( i \) of layers do
7: if \( i \) \( = \) 0 then
8: \( T_{i2} \leftarrow \text{time} = \text{time} + 1, \text{Propagate}_i(t) \)
9: else
10: \( T_{(i+1)(i+2)} \leftarrow \text{Propagate}_{i+1}(t) \)
11: end if
12: \( \text{node}_{i+2} \leftarrow L[i+1] \)
13: end for
14: \( T_{(n+1)1} \leftarrow t < \text{time} \& \& \text{time} < t_0 \)
15: \( T_{(n+1)(n+2)} \leftarrow \text{Guard()} \)
16: return \( A_{\text{RNN}} \)

ant in \( \text{Output} \_\text{Layer} \) (i.e., \( t \leq 3 \)). Finally, we assign \( \text{Guard()} \) to transition from \( \text{Output} \_\text{Layer} \) to \( \text{Success} \).

For LSTM [5], we model the \( \text{forgetgate}, \text{inputgate} \) and \( \text{outputgate} \) into automaton. Using idea similar to Algorithm 3, update processes within LSTM can be abstracted
and assigned to corresponding transitions between states. And we assign time update assignment \( \text{time} = \text{time} + 1 \) to transition between state \( \text{Wait} \_\text{for} \_\text{input} \) and state \( \text{Forget} \_\text{gate} \_\text{and} \_\text{input} \_\text{gate} \_\text{updated} \).
Guard \( t \leq \text{time} \& \& \text{time} < t_0 \), same as Algorithm 3 is constructed to control the number of rounds. Actually, Guard \( t \leq \text{time} \& \& \text{time} < t_0 \) can be reduced to \( \text{time} < t_0 \). The We model a two-layer LSTM as the running example. The Automaton obtained is shown in Fig. 5.

V. FORMAL VERIFICATION

In this section, we show how to verify the properties of DNNs. Intuitively, the properties of DNNs can be converted into the properties of automata. These properties can be verified by existing model checkers such as UPPAAL.

For FNN and CNN, because they have similar structures, methods of transforming their properties are inspired by the same idea. To be specific, we use the \( \text{Guard()} \) function to control the operation of the automaton and encode the properties of FNN (or CNN) into the condition of if-else statement. The \( \text{Guard()} \) is designed in such a way that it returns True when the properties are as expected, and False otherwise. In other words, if and only if FNN (or CNN) meets the property, the automaton is deadlock-free, otherwise the automaton will deadlock (i.e., stop on state \( \text{Output} \_\text{layer} \)). As a result, we are able to verify the properties of FNN or CNN by using UPPAAL to check whether the system is deadlock-free with the query \( A[ ] \) not deadlock.

For Vanilla RNN and LSTM, we construct \( \text{Guard()} \) in a similar way. \( \text{Guard()} \) returns True if the property of the RNN is as expected. Otherwise, \( \text{Guard()} \) returns False. Naturally, when the RNN meets the expected property, the automaton always returns to the state \( \text{Success} \) for both Vanilla RNN and LSTM. Otherwise, the automaton eventually stops in the state \( \text{Output} \_\text{layer} \). With the help of the model checker, we can check the property “the automaton can always reach the state \( \text{Success} \)” with the query \( E <> \text{Process} \_\text{Success} \).

The traditional definition for local robustness based on distance is ”for every input \( x_1 \) and \( x_2 \) such that \( |x_1 - x_2| < \delta \), if the network is able to assign the same label to \( x_1 \) and \( x_2 \), then the network is robust.” In UPPAAL, we can assign arbitrary value to input relying on demand. Thus, when exploring the robustness of the DNN, we could change inputs’ value and observe whether the corresponding property (i.e., the output situation), could remain unchanged. If so, we could say the local robustness of the DNN for specific input could be proved.

VI. CASE STUDIES

To examine the effectiveness of DeepAuto, we conduct case studies on both primitive operations and medium-sized DNNs. In our experiments, we extract the parameters required by modeling from the DNNs, and then use DeepAuto to model and verify them. To show the practicability of DeepAuto, there are two critical points that we need to highlight: (1) We can extract weights of trained DNNs based on Keras with TensorFlow. (2) The properties of DNNs can be converted into the properties of automata with the aid of data flow.

We start with a small-scale experiment in which we train FNNs which are capable of simulating \( \wedge \) (and), \( \vee \) (or), \( \sim \) (not) and the Exclusive-OR gate. They are the four primitive operations the DL seminal work [8] uses neural networks to simulate. With DeepAuto, the weights of trained FNN can be automatically imported into UPPAAL, and then the formal models of these FNNs can be given automatically. Now we can obtain formal model of FNNs which are capable of simulating
all propositional forms with $\land$, $\lor$ and $\neg$. We can also model trained CNN and trained RNN as automaton automatically.

A standardized procedure for modeling and verifying a trained DNN is presented in Fig. 6. Following the procedure, we can model and verify the DNN before deployment, and verify whether the DNN is running according to its expected properties formally. When the DNN does not meet its expected properties, the automata can be used to observe the data flow inside the DNN, so as to point the way for debugging.

Fig. 6. The Standardized Procedure for Modeling and Verifying a Trained DNN

Theoretically, DeepAuto can be scaled up to verify practical DNNs. However, UPPAAL doesn’t accept real arithmetic. The weights of DNNs which we study in experiments are INT. We show DeepAuto can be scaled up to verify medium-sized DNNs. Several critical properties, as shown in Table I, can be verified based on DeepAuto framework. Property $P_3$ states that if the input is not perturbed by adversarial attacks, the output of the DNN will always be as expected. Property $P_2$ deals with the problem whether the output will be affected when the input is perturbed. Property $P_3$ “when we modify the internal weights, will the output be as expected?” probes into the interpretability of DNNs. As presented in Table I, we are able to verify all these properties based on the DeepAuto framework. The details can be found at https://github.com/Yuteng-Lu/UPPAAL

The concrete constructing and verifying processes of these three properties are as follows: (1) For property $P_1$, the expected relationship of input data and output data could be written in the Guard(). (2) For property $P_2$, we could change input’s value based on adversarial attack’s behavior. After generating the perturbed input, we could check the properties of our formal model. If the properties change, we say the output will be affected when the input is perturbed. (3) For property $P_3$, we can alter the formal model to modify the internal weights. After getting the formal model with modified weights, we could check its properties to indirectly check $P_3$.

VII. CONCLUSION

In this paper, we propose the modeling and formal verification framework DeepAuto for DL systems. The UPPAAL model checker is used to implement DeepAuto. The experiments show that DeepAuto can dramatically guide modeling and verification procedure of DNNs, and the method covers the three most common DNN structures: FNNs, CNNs and RNNs. In addition, our work illustrates that there is a connection between the states of the timed automaton and the DNN’s neurons. On the other hand, DeepAuto is actually a white-box verification framework. How can we do formal verification without all internal information (i.e., black-box or gray-box) will be considered in future work.

ACKNOWLEDGMENTS.

This work was partially supported by the Guangdong Science and Technology Department (Grant no. 2018B010107004) and the National Natural Science Foundation of China under grant no. 61772038 and 61532019.

REFERENCES


<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRITICAL PROPERTIES</td>
</tr>
<tr>
<td>Specific Properties</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>$P_3$ When the input is not perturbed, the output is as expected.</td>
</tr>
<tr>
<td>$P_2$ Whether the output will be affect when the input is perturbed.</td>
</tr>
<tr>
<td>$P_3$ What happens to the output when the internal weights are changed.</td>
</tr>
</tbody>
</table>
Evaluating the Impact of Vaccination on COVID-19 Using Model Checking

Xin Li
Shanghai Key Laboratory of Trustworthy Computing
East China Normal University, Shanghai, China
xli@sei.ecnu.edu.cn

Abstract—Since last year, COVID-19 has become a global issue and brings crisis in nearly every aspect. At the end of last year, COVID-19 vaccines finished developing and began to be large-scale administered. However, the impact of vaccination on COVID-19 is not clear yet. The current vaccine manufacturing capacity is highly insufficient, thus, it is urgent to implement an optimized vaccine prioritization strategy. To shed light on these vaccination issues in our study, we used a fluid model checking method to track the trends of COVID-19 dynamically. First, we proposed a vaccine-related epidemiological model, the SEIHRV model, to investigate the vaccine’s impact on the pandemic based on previous studies. Then we embraced a fluid model checking approach to evaluate this SEIHRV model. Compared with the compartmental and agent-based model methods, our approach made a trade-off between speed and accuracy. Lastly, we took several exceptional scenarios into account based on the current COVID-19 situations, including the individual’s daily activities, hospital capacity, vaccine prioritization strategy, the variant virus. Our work applied this fluid model checking method to COVID-19 studies, which demonstrates the up-to-date computing method can combine with social concerns and deal with practical problems from an innovative perspective.

Index Terms—COVID-19; Vaccine; SEIHRV model; Fluid model checking; Continuous-time Markov-chain

I. INTRODUCTION

In the past year of 2020, coronavirus disease 2019 (COVID-19) pandemic which is caused by a severe acute respiratory syndrome coronavirus type 2 (SARS-CoV-2) has swept the world, seriously affected and even threatened all aspects of people’s lives. As of March 9, 2021, more than 117 million COVID-19 cases have been reported worldwide, of which death cases exceeded 2.6 million (data from Center for Systems Science and Engineering at Johns Hopkins University [1]). In order to reduce the infection rate and mortality rate of COVID-19, governments all around the world have implemented a series of lockdown policies. Researchers in different domains from different countries also actively participate in the COVID-19 studies, especially, in developing the COVID-19 vaccine.

Many COVID-19 vaccine projects started since the beginning of 2020, vaccine developed by BioNTech/Pfizer first obtained the emergency use authorization by FDA (U.S. Food and Drug Administration) and European Union on December of 2020. Subsequently, another vaccine developed by Oxford and AstraZeneca [2] was also approved by several European countries. The emergence of vaccines brings the hope of defeating COVID-19 to people. Ideally, if the majority of the population obtain protection from vaccines, COVID-19 reproduction rate would decrease, and it would disappear or be inactivated in the end.

However, there are still some challenges in achieving herd immunity through vaccination. The first challenge is the manufacturing capability. Although vaccine manufacturers have accelerated the large-scale production of vaccines, so far, there are only 300 million doses have been administered worldwide [3]. Due to this situation, countries have to implement various priority strategies to ensure that priority groups (such as the elderly, medical staff, etc.) can be first administered. Nevertheless, these strategies might not be the most efficient one [4]. Second, vaccine efficacy is not completely clear. According to statistics, BioNTech/Pfizer vaccines have an efficacy rate close to 95 percent [5], while Oxford-AstraZeneca vaccines have 82.4 percent for two doses separated by 12 weeks [6]. Third, newly emerged variants SARS-CoV-2 called B.1.1.7 (from the United Kingdom) and B.1.351 (from South Africa) induce increased risk of death and infection, but how they affect vaccine efficacy is still an open question.

Since various factors have brought great uncertainty to the COVID-19 pandemic and the vaccine efficacy, a method that can dynamically monitor the vaccine efficacy is urgently required. Jensen and colleagues used the UPPAAL SMC tool to not only simulate the overall trend of COVID-19 pandemic, but also dynamically track the individual’s infection rate and other COVID-19 indicators [7]. Based on their work, we also applied UPPAAL here to check how the Susceptible-Exposed-Infected-Hospitalized-Recovered Model (SEIHR model [8]) simulates the COVID-19 pandemic after the large-scale vaccine administration. Here, we considered the latest COVID situations, including the vaccine prioritization strategy [4], the age distribution of COVID cases [9], vaccine efficacy, and vaccine performance against variant viruses. In this way, we can get a comprehensive understanding of how this SEIHR model simulating this pandemic in various situations.

The contributions of this paper are as follows:

- We proposed a vaccine-related epidemiological model to investigate the vaccine’s effects on the COVID-19 pandemic.
- We embraced the fluid model checking to analyze the infectious disease model. Compared with the compartmental and the agent-based model method, our approach...

DOI reference number: 10.18293/SEKE2021-131
makes a balance between speed and accuracy.

- We demonstrated some complicated situations could be efficiently simulated by our model, such as the individual case situations, hospital capacity, and vaccine prioritization strategy. This approach bridges the gaps between up-to-date computing methods and social concerns.

This paper is organized into five sections: Section II introduces the background of epidemiological models and a stochastic model checking method, while Section III describes the widely-used basic reproduction number and our SEIHRV model. Section IV emphasizes our experiment and discussion. Section V draws the conclusion.

II. BACKGROUND

A. The Compartmental Model in Epidemiology

In epidemiology, Kermack and McKendrick proposed a dynamic compartmental SIR model to simulate the influences of an infectious disease [8]. The original SIR model divided individuals into three groups depending on their physical states (compartments): susceptible (S), infectious(I), and recovered (R) group. It assumes that all individuals are initially susceptible, while all recovered individuals will have antibodies and will not be re-infected again. The dynamical flow of this SIR model can be represented as follows:

\[ \frac{dS}{dt} = -\frac{\beta SI}{N} \quad \frac{dI}{dt} = \frac{\beta SI}{N} - \gamma I \quad \frac{dR}{dt} = \gamma I \]

The establishment of this dynamical model sheds light on theoretical and quantitative analysis of infectious diseases [10]. It is based on the disease’s occurrence, the spread, the transmission rules, and related social factors. Through qualitative, quantitative analysis, and numerical simulation, it is possible to track where the disease comes from, monitor some key elements, and even predict the disease trends.

Using the continuous-time ordinary differential equation (ODE) method, this traditional compartmental model can easily simulate the general population’s epidemic development. Nevertheless, it does not allow the prediction for individuals, how compartment changed with individuals’ activities. Moreover, it is impossible to describe some exceptional cases, like the variant COVID-19 virus and the super spreaders. To simulate these complicated cases, a fluid model checking is necessary.

B. Stochastic Model Checking

Stochastic model checking is an extension of classical model checking theory, serves as an automatic model-based formal verification of stochastic systems [11]. In recent years, stochastic model testing has attracted a lot of attention in the formal verification domain and has made significant progress. It has been applied to the verification of probabilistic programs, system performance analyses, communication protocol reliability analyses, service quality optimization of service processes, and even computational biology [12]. Typically, the continuous-time Markov chain (CTMC) [13] is used to represent models in stochastic model checking, CTMC is a tuple \((S, R, \pi_0)\) where:

- \(S\) is the set of states,
- \(R : S \times S \rightarrow \mathbb{R} \geq 0\) is the transition rate matrix, and
- \(\pi_0\) is the initial distribution.

We defined the exit rate of \(s\) state \(s \in S\) as

\[ E(s) = \sum_{s' \in S} R(s, s') \]

The embedded DTMC coincides on \(S\) and \(\pi_0\) but has the transition probability matrix \(E\). It can be defined as:

\[ E(s, s') = \begin{cases} \frac{R(s, s')}{E(s)} & \text{if } E(s) > 0 \\ 0 & \text{if } E(s) = 0 \land s \neq s', \text{and} \\ 1 & \text{if } E(s) = 0 \land s = s'. \end{cases} \]

The probability measure for a CTMC \((S, R, \pi_0)\) is induced by the measure for cylinder sets \(P(C(s_0/0 \ldots s_n))\) defined as:

\[ \pi_0(s) \prod_{0 \leq i < n} E(s_i, s_{i+1}) \left(e^{-E(s_i)\inf l_i} - e^{-E(s_i)\sup l_i}\right). \]

The stochastic model checking algorithm combines the classical model checking algorithm and linear equation system solution or linear programming algorithm. Its operation deals with the probability vector about the state instead of the bit vector in the classical model checking. However, the stochastic model checking is faced with a more challenging problem, the state-space explosion problem. It is still not clear how to mitigate the state explosion problem for large-scale stochastic systems and apply stochastic model testing to the quantitative verification and analysis of such systems, even Turing Award winner Clarke listed this as a crucial direction for future research in model testing in his Turing Lecture [14]. Researchers recently found that fluid model checking, a combination of ODE and CTMC, can efficiently mitigate the state-space explosion [15].

III. BASIC REPRODUCTION NUMBER AND SEIHRV MODEL

MacDonald and colleagues established a mathematical model on the spread of an epidemic based on malaria studies [16]. There is a crucial indicator in their model, the basic reproduction number \(R_0\). \(R_0\) that greater than 0 indicates this epidemic will continue to spread; otherwise, this epidemic will eventually be eliminated. The reproduction number has been widely used in the epidemiological domain, and now it becomes one of the most valuable indicator in COVID-19 studies [17].
In our paper, a variant of the SIR model: the SEIHRV model was defined. We also assumed all individuals are initially susceptible (S), they become exposed (E) once they contact with an infectious person, the transition rate can be denoted as $\lambda$. After a latency period ($d_{le}$ days), the exposed individual will transition to an infectious individual (I). Those infectious people with severe symptoms for few days ($d_{lH}$) are accepted to the hospital (H) (HR: hospitalization rate), the majority of them will recover (R) after several days ($d_{l}$) treatment, a small proportion (IFR: infection fatality rate) would die even been treated. Meanwhile, most infectious people will spend some days ($d_{lH}$) to eventually recover (R) by relying on their self-immunity. In this model, $\lambda$ is obtained by:

$$\lambda = u \sum_j c_j \frac{I_j}{N - \Omega}$$

where $u$ is the probability of a successful transmission from an infectious contact and the $c_j$ is the number of individuals that a susceptible individual contacts per day. $I/(N - \Omega)$ refers to the probability that a random individual is infectious, where $N$ is the total population number, $I$, $\Omega$ is the infectious number and death number, respectively. $\lambda$ can also predict basic reproduction number $R_0$.

![Schematic diagram of a dynamic SEIHRV model with vaccination.](image)

**Fig. 1. Schematic diagram of a dynamic SEIHRV model with vaccination.**

To evaluate the development of the COVID-19 pandemic after large-scale vaccine administration, we added vaccination effect in the original SEIHR model and named the new model as SEIHRV model. We divided the whole population into three groups as previous work did [4]: people were vaccinated and effectivly protected by the vaccine, unvaccinated people (can be vaccinated), and people ineligible for vaccination, which includes vaccine-hesitant people, people with positive serotest result or vaccinated people without protection. The schematic diagram is shown in Fig. 1. Whether an individual can be administrated or whether a vaccinated person is perfectly protected largely depend on the serological test results. However, the serological test is not entirely accurate. To illustrate this constraint, we also took the sensitivity ($se$) and the specificity ($sp$) of the serological test into consideration. $\alpha$ in this model represents the vaccine rollout rate in susceptible and exposed populations, and $\alpha = \frac{n_{vax}}{[(S + E)sp + R(1 - se)]}$, where $n_{vax}$ indicates the amount of vaccines to be rolled out in a single day. $v_e$ denotes the vaccine efficacy. A small proportion of the susceptible people and the exposed population with negative serotest results can be vaccinated ($\alpha(sp)$) per day. Some of them are well protected by the vaccine and will be members of $V$ or $E_v$ group, the rest of the people who are ineligible for vaccination or without vaccine protection will be grouped into $S_x$ and $E_x$. The following equations describe the compartment-to-compartment relationships:

$$S \to S_v = \frac{n_{vax}}{(S + E)sp + R(1 - se)}(S(sp))$$
$$E \to E_v = \frac{n_{vax}}{(S + E)sp + R(1 - se)}(E(sp))$$
$$R \to R_v = \frac{n_{vax}}{(S + E)sp + R(1 - se)}(R(1 - sp))$$
$$S \to S_x = \frac{n_{vax}}{(S + E)sp + R(1 - se)}(S(1 - sp))$$

**IV. Experiment and Discussion**

In this section, we first built a continuous-time ordinary differential equation model of SEIHRV, and simulated the 365-day dynamic model corresponding to the first-year vaccination phase. Then we used the continuous-time Markov chain models since they have better performance than ODE if we take individuals’ behaviors and decisions into consideration, such as staying at home or wearing a mask. Usually, such models require a large volume of individual activity data and a considerable amount of computational resources.

To improve these, we got the inspirations from the fluid model checking method, combined the ODE and CTMC, to simulate the COVID-19 epidemic. By doing this, the system required less time in performing verification significantly. Thus, we can not only use ODE to describe the population epidemic development, but also use CTMC to describe the individuals’ status. Our simulations indicate lockdown and conditional lockdown are the best options for reducing infection fatality rate and virus transmission rate.

Additionally, we simulated the emerging cases caused by the variant and emphasized how the variant affects hospitalization rate and virus transmission rate.

A. Ordinary differential equation models

In the first part, we established the ODE model of SEIHRV in UPPAAL SMC [18]. All individuals in this model are initially susceptible unless they have been effectively vaccinated or have already got natural immunity. The recovered individuals are no longer infectious in the simulation period (up to 365 days) and cannot be re-infected. Details of SEIHRV model can be found in Section III.

As shown in Fig. 2, only one state was required for one individual to build the above ODE model system in UPPAAL SMC, and all of the populations were represented by using the variable CLOCK in UPPAAL SMC.
We hypothesized a town with a 10,000 population in our simulation, the basic reproduction number of this town is 1.3, and 1% people have been infected. Other demographic data, such as infection and fatality rate, were obtained from CDC [1]. Meanwhile, we assumed the vaccine rollout rate is 0.2% per day, and the vaccine efficacy is 95%, which means vaccines are not one hundred percent efficient. The specificity and sensitivity value of a serological test is 0.97 and 0.99.

**TABLE I**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epidemic trend</td>
<td>simulate ([&lt;=365]{S,E,I,H,R,V,D})</td>
</tr>
</tbody>
</table>

Next, a query was described in UPPAAL SMC. As shown in TABLE I, we counted the susceptible, exposed, infectious, hospitalized, recovered, and dead people within 365 days and analyzed COVID-19 trends.

Simulation results indicate this epidemic is hard to control at the very beginning of the vaccination. However, the situation will become better once the number of vaccinated population increases, the transmission rate will simultaneously decrease (Fig. 3). Around 60 days after vaccination, the susceptible line and the recovered line will intersect, the inflection point of the epidemic appear, which means the epidemic is gradually under-controlled.

**B. Continuous-time Markov chain models and Fluid model checking**

Although the ODE model in the previous section can describe the epidemic trends at the population level, it cannot describe individuals’ behavior and incorporate some other details, like the age-dependent infection rate and fatality rate, the lockdown, etc. Therefore, we built the CTMC model of SEIHRV here to involve detailed information (Fig. 5):

In this model, each location has one corresponding exponential distribution function, and different weights were given to different edges. For example, in Fig. 5, location \( I \) has different probabilities to location \( H \) or \( L \). Initially, we assumed a CTMC could describe an individual’s status, and used the 10,000 population case to test it. However, we found this operation occupied massive computing resources, since 10,000 CTMC models are required to generate based on the template and simultaneously run. Moreover, the state-space of the system increased exponentially as the number of people increased, and the state-space explosion problem appeared.

Therefore, in order to efficiently model these data with individual information, we adopted the fluid model checking method in our experiment, abstracted the population epidemic situation, and only focused on the target groups. Combination of the ODE model and the CTMC model in UPPAAL SMC can significantly reduce the memory and time required for simulation and verification.

We simulated a lockdown scenario here and found transmission significantly decreased in the lockdown period (Fig. 4(a)). Furthermore, we selected 30 specific individuals in this model and analyzed their health conditions (Fig. 4(b)). If their health conditions varied largely, their personal curves would also be fluctuating. The above simulation took a long time if we only used the CTMC model. However, the simulation time shortened to one-tenth if the CTMC model was used to represent 30 target individuals and others are modeled by ODE. Thus, the fluid model checking significantly reduced the memory and time required for simulation and verification.
C. Hospital capacity and Variant of SARS-CoV-2

A variant of the SARS-CoV-2 was first identified in UK in September 2020, and then it spreads around the world in the next few months. Studies show this variant has a 40-70% higher transmission rate than the original virus [19]. By far, it has been found in more than 100 countries and takes Germany as an example, the variant infected nearly 55% of the cases. Studies demonstrated this variant could also induce a 30-100% higher infection fatality rate than previous viruses.

To include this variant in our model, we accordingly modified our parameters. In the updated model, 55% of patients were infected by the new variant. Furthermore, due to the high infection rate and fatality rate caused by this variant, we addressed hospital capacity here. Once the number of patients with severe symptoms exceeds hospitals’ capacity, the fatality rate would rise rapidly. So we had the following query in TABLE II, the query “Hospitalized trend” can be explained as how the number of hospitalized population changes.

One hundred hospitalized cases were simulated here, data shows hospitalization peak occurs at around 40 days after vaccination, and the number of hospitalized patients is about 60 (Fig. 6). Those simulation data can help medical institutions monitor the hospitalization load dynamically and make complete preparations in advance.

To get more details, we used the query “Hospital capacity” to calculate the probability of hospitalized cases exceed 60. As shown in Fig. 7, the range is [0.0125218,0.112309] with confidence 0.95. Thus, we suggest medical institutions to pay sufficient attention to this variant, it could bring vast pressure on the medical system.

D. Vaccine prioritization strategy

At the beginning of large-scale vaccination, vaccines’ manufacturing capability is limited and it is still hard to achieve herd immunity, optimizing the vaccine prioritization strategy has become a crucial issue. Most prioritization strategy is age-dependent since infection rate and infection fatality rate are closely associated with ages. In this section, we applied the
fluid checking model to analyze various vaccine prioritization strategies.

Here, three different vaccines were used: Pfizer-BioNTech, Moderna, and Oxford/AstraZeneca. Their efficacies are 95% and 94.1% and 82.4%, respectively. We used the age demographic data from UN [20] to estimate the age distribution, and grouped individuals by their ages with 10-year increment steps. Some other age-stratified values from previous literature [9] served as model parameters here, which include infection rate, fatality rate, hospitalization rate, and recovery period.

Four age-dependent vaccine prioritization strategies: young people under 20 years old, middle-aged and young adults between 20 and 49 years old, old people over 60 years old, and all people over 20 years old, were simulated by using our previous approach.

We assumed all remaining individuals have equal priority once the vaccination for the priority group has been completed. Here we present simulations of 20-49 (Fig. 8(b)) and 60+ (Fig. 8 (a)) prioritization strategy (Pfizer-BioNTech).

![Simulation results](image)

Fig. 8. Simulation results

20-49 vaccine prioritization strategy effectively lowers down the infection rate and transmission rate (Fig. 8(b)). Those results are reasonable because individuals at that age tend to have more social activities, most of them work everyday and are more likely to contact infectious individuals. Meanwhile, 60+ strategy can effective reduce the fatality rate (Fig. 8 (a)) mainly because old people usually have higher hospitalization rates and fatality rates once they are infected. Since every vaccine prioritization strategy has its own benefits, it is crucial to make a trade-off between different COVID-19 vaccine prioritization strategies.

V. CONCLUSION

In this paper, we used a fluid model checking method to conduct a series of studies on the current COVID-19 pandemic. First, we proposed a vaccine-related epidemiological model: the SEIHRV model. With this model, we are able to investigate the vaccination impact on the current pandemic more rapidly, efficiently, and flexibly. Second, the fluid model checking method was embraced to model, simulate, verify, and analyze this SEIHRV model. We elaborated this method in detail and compared it with the traditional compartmental model method. Results show it performed better in complicated scenario simulations. Moreover, this method is adjustable on different individual’s specific conditions and can provide more detailed information. In comparison with the agent-based method, fluid model checking can effectively reduce computing resources' requirements and lessen the state space explosion. Finally, based on the true-to-life situations, we simulated the cases individually, also how the hospital capacity, variant virus, and vaccine prioritization strategy affect the COVID-19 trends. In conclusion, this approach provide a more practical and efficient way to simulate COVID-19 development in complicated scenarios, especially after the large-scale vaccine administration. Correspondingly, its simulation results can provide more comprehensive and detailed information on COVID-19.

REFERENCES

A Novel Approach of CTL Model Checking Based on Probe Machine

Dong Wang†, Jing Liu∗†, Jin Xu‡, Haiying Sun†, Jiexiang Kang§
† Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
‡ Key Laboratory of High Confidence Software Technologies, Peking University, Beijing, China
§ China Aeronautical Radio Electronics Research Institute, Shanghai, China

Abstract—Model checking has established as an effective method for automatic system analysis and verification. It is making its way into many domains and methodologies. However, the state space may be extremely large for many practical systems, and this is a major limitation for state-space search algorithms in model checking. We have proposed a novel computing model called probe machine in 2016, which is a fully parallel computing model. In comparison to the Turing machine, it can solve the graph search problems efficiently, which can overcome the existing model checking limitations. In this paper, we propose a novel approach to perform Computation Tree Logic (CTL) model checking based on the mathematical model of probe machine, which can verify all CTL properties. It can greatly reduce the verification time for systems with large state space. We develop a model checker called CTL2PROBE based on our approach and the experimental results show that our approach is better than NuSMV.

Index Terms—Model Checking, Probe Machine, Graph Search, Computation Tree Logic

I. INTRODUCTION

Model Checking is a technology that can verify whether a system satisfies the given property automatically [1]. The way of verification is exploring all possible system states in a brute-force manner [2]. In this way, it is a real challenge to examine the largest possible state spaces that can be treated with current means, i.e. processors and memories. Therefore, the chief limitation is the state explosion problem where the size of the global state graph grows exponentially with the size of the program itself [3].

In order to deal with the state explosion problem, many approaches have been proposed. Ordered Binary Decision Diagram (OBDD) is proposed by McMillan to represent state space, improving the scale of verified system [4], [5]. Another successful technique for dealing with state explosion is based on the partial order reduction [6], [7]. It exploits the independence of concurrently executed events. Although symbolic representations and partial order reduction have greatly increased the size of the systems that can be verified, many realistic systems are still too large to be handled. Thus, some researchers have turned their attention to other novel computation architectures like molecules and DNA to break through the limitations of Turing machines.

In 2006, Turing Award winner Allen Emerson use DNA molecules to conduct CTL model checking for the first time [8]. In this work, he proposes a DNA-computing-based method and designs a checking algorithm for CTL formula EFp where Kripke structures are used to model the system and the states and transitions of the model are encoded into DNA strands. It permits a very compact representation of extremely large state graphs. For example, a graph of size 10^{16} states can be represented within 0.01 liter of DNA. Therefore, the parallelism of DNA computing and the vast storage of DNA molecules provide the opportunity to break the state space limitation on traditional electronic computers.

Probe machine is a mathematical model proposed by Xu in 2016 [9]. It can be implemented by using nano-DNA probe technologies. It is a fully parallel computing model in the sense that it can simultaneously process multiple pairs of data, rather than sequentially process every pair of linearly adjacent data. Many NP-complete problems, i.e., the graph coloring, Hamilton cycle problems, traveling salesman problem [10], and postman problem [11] have been solved based on the probe machine. Probe machine can enumerate all solutions to these problems by only one probe operation.

Similarly, the way of CTL model checking is exploring all possible system states in a brute-force manner and finding a path satisfying the given property. Therefore, we propose a novel method to perform CTL model checking based on the probe machine. Compared to traditional model checkers, our approach can relieve the state explosion problem and reduce the verification time.

In summary, this paper makes the following contributions:

- We design a mapping algorithm to transform the model of Kripke structure into the data library and probe library that can run directly on the probe machine.
- We develop a model checker called CTL2PROBE to simulate the probe machine, which takes the model as input and obtains all feasible paths or counterexamples.
- We conduct several experiments based on different numbers of states. Compared to NuSMV, the experiment results prove the feasibility and efficiency of our approach.

The rest of this paper is organized as follows. Section II briefly introduces CTL model checking and the concept of probe machine. Section III presents our approach and provides complexity analysis. Section IV introduces the model checker...
II. PRELIMINARIES

A. CTL Model Checking

A Kripke structure is a variation of the transition system, originally proposed by Saul Kripke [12], used in model checking [13] to represent the behavior of a system. It consists of a graph whose nodes represent the reachable states of the system and whose edges represent state transitions, together with a labeling function which maps each node to a set of properties that hold in the corresponding state.

A Kripke structure is defined as a four-tuple

\[ M = \{ S, I, R, L \} \]

- a finite set of states \( S \),
- a set of initial states \( I \subseteq S \),
- a transition relation \( R \subseteq S \times S \),
- a labeling function \( L : S \to 2^{AP} \).

Computation Temporal Logics properties are traditionally interpreted in terms of Kripke structures. Clarke has proved that any CTL formula can be expressed in terms of \( \neg, \lor, \mathbf{EX}, \mathbf{EU} \) and \( \mathbf{EG} \) [13]. Thus, it is sufficient to be able to handle six cases, depending on whether \( g \) is atomic or has one of the following forms: \( \neg f_1, f_1 \lor f_2, \mathbf{EX} f_1, \mathbf{E}[f_1 \lor f_2] \) or \( \mathbf{EG} f_1 \). These CTL formulas describe the following properties.

\[
\begin{align*}
M, s \models \neg f_1 & \iff M, s \not\models f_1 \\
M, s \models f_1 \lor f_2 & \iff M, s \models f_1 \text{ or } M, s \models f_2 \\
M, s \models \mathbf{EX} f_1 & \iff \text{there exists a state } t \text{ such that } R(s, t) \text{ and } M, t \models f_1 \\
M, s \models \mathbf{E}[f_1 \lor f_2] & \iff \text{there exists an infinite path } \pi \text{ starting at } s \text{ and there exists a } k \geq 0 \text{ such that } M, s_k \models f_2 \text{ and for all } 0 \leq j < k, M, s_j \models f_1 \\
M, s \models \mathbf{EG} f_1 & \iff \text{there exists an infinite path } \pi \text{ starting at } s \text{ such that for all } i \geq 0, M, s_i \models f_1
\end{align*}
\]

B. Probe Machine

Probe Machine (PM) is defined as a nine-tuple

\[ PM = (X, Y, \sigma_1, \sigma_2, \tau, \lambda, \eta, Q, C) \]

where the nine components denote the data library \((X)\), probe library \((Y)\), data controller \((\sigma_1)\), probe controller \((\sigma_2)\), probe operation \((\tau)\), computing platform \((\lambda)\), detector \((\eta)\), true solution storage \((Q)\), and residue collector \((C)\). The following will introduce four main components: data library, probe library, probe operation, and computing platform.

- The data library \(X\) is viewed as a set of \(n\) elements, denoted by \(X = \{x_1, x_2, ..., x_n\}\). Each data \(x_i\) contains a body and \(p\) types of data fibers. Data \(x_i\) is defined as \(x_i = \{x^1_i, x^2_i, ..., x^p_i\}\)

- The probe in the probe machine is defined as a tool to find two data and implement some operations (e.g., connective and transitive operations) between them. Formally, let \(x^j_i\) and \(x^m_i\) be two types of data fibers. The probe between \(x^j_i\) and \(x^m_i\) denoted as \(\tau_i^{j \rightarrow m}\). A connective probe, denoted as \(\tau_i\), refers to a probe \(\tau_i^{j \rightarrow m}\) connecting two target data fibers \(x^j_i\) and \(x^m_i\), forming a high-order aggregation.

- A probe operation \(\tau\) is a process of executing many probe operations simultaneously.

- The computing platform, denoted by \(\lambda\), is an environment to conduct probe operations \(\tau\). It helps probes rapidly find the target data fibers and then conduct probe operations. High cohesiveness, threshold property, and uniqueness are the fundamental functions of the computing platform.

- High cohesiveness is the rule that high-order aggregation data are given higher priority than low-order aggregation to be executed probe operation.

- Threshold property limit that the size of two aggregations for probe operation must not exceed the size of the graph itself.

- Uniqueness is the rule that there is at most one data for each type that an aggregation contains.

III. MODEL CHECKING BASED ON PROBE MACHINE

This section is concerned with CTL model checking on the probe machine. We first introduce the procedures of model checking on the probe machine. Construction of the data library and probe library are two main steps. Subsequently, we propose the methods to construct data library and design methods for probe library of \(\mathbf{EG}\) and \(\mathbf{EU}\) respectively. In the end, we analyze the time complexity for the designed approach.

A. Procedures of Model Checking on Probe Machine

Xu solve the graph coloring problems with connective probes that connect two adjacent vertexes of different colors [9]. Inspired by the previous work, we propose a graph search mechanism for model checking. Here, we use a probe to connect two adjacent states in the graph which satisfying the CTL property. Each subpath is considered as data to be probed. In this way, each subpath continuously grows after each probe operation until the initial state is included in the path.

The procedures of model checking on the probe machine are divided into four steps as follows

- Construction of the data library \(X\).
- Establishment of the probe library \(Y\) according to the CTL property.
- Implement the probe operation \(\tau(X, Y)\) in the computing platform \(\lambda\), producing a large number of solutions.
- Find the true paths \(T\) of all solutions.

Specially, Clarke has proved that any CTL formula can be expressed in terms of \(\neg, \lor, \mathbf{EX}, \mathbf{EU}\) and \(\mathbf{EG}\) [13]. Thus, it is sufficient to be able to handle six cases. We usually check \(L(s)\) to verify \(M, s \models \neg f_1, M, s \models f_1 \lor f_2\) and traverse the successors of the state \(s\) to check \(M, s \models \mathbf{EX} f_1\). These
three verifications can be solved within a short time, so we don’t need some particular methods on the probe machine to check them. Therefore, only two algorithms are needed for \(\text{EG}\) and \(\text{EU}\). And the following will respectively introduce how to build the data library and probe library for \(M, s \models \text{EG} f_1\) and \(M, s \models \text{EU} [f_1 U f_2]\) respectively.

### B. Construction of Data Library

Let a graph \(G\) mean a Kripke structure, \(M = \{S, I, R, L\}\). We denote \(V(G)\) and \(E(G)\) as the sets of states and transitions of \(G\) respectively. \(G\) has a set of nodes \(V(G) = \{v_1, \ldots, v_n\}\) and a set of edges \(E = \{e_1, \ldots, e_p\}\), as well as \(L(v_i) \in \{f_1, \ldots, f_n\}\) holds for each \(v_i\).

In addition, we denote \(\text{Pre}(v_i)\) and \(\text{Suc}(v_i)\) as the sets of predecessor nodes and successor nodes of the vertex \(v_i\), \(E(v_i)\) is defined as the set of edges out of the vertex \(v_i\). Let \(E^2(v_i)\) be the set of all directed two-paths with internal vertex \(v_i\). Formally

\[
E^2(v_i) = \{ v_i v_i v_r \triangleq x_{li} | v_l \in \text{Pre}(v_i), v_r \in \text{Suc}(v_i) \}
\]

Data \(x_{li}\) are defined according to (1). Let’s take an example[see Fig. 1(a)], it is a simple Kripke structure with 3 states and 2 transitions. Every state is considered as data with its transition edge, such as \(E^2(v_1) = \{x_012\}\) [see Fig. 1(c)].

Based on \(E^2(v_i)\), we construct the data library \(X\) of the connective probe machine as follows:

\[
X = \bigcup_{i=1}^{n} E^2(v_i) = \bigcup_{i=1}^{n} \{ x_{li} | v_l \in \text{Pre}(v_i), v_r \in \text{Suc}(v_i) \}
\]

where \(x_{li}\) has exactly two types of data fibers, the left one is \(x_{li}\) and the right one is \(x_{ri}\). Typically, initial states have no predecessor and final states have no successor, so we define \(\omega\) as empty. The data \(x_{w01}\) of the initial state \(s_0\) is shown in Fig. 1(b).

In fact, each data represent a path and a \(n\)-aggregation data represents a path of length \(n\). For example, data \(x_{012}\) represents a one-length path, \(\pi = v_1\) [see Fig. 1(c)], and 0 and 2 respectively represent the possible node connecting this path. Similarly, a two-aggregation data \(x_{w012}\) represents a two-length path, \(\pi = v_0, v_1\) [see Fig. 1(e)].

### C. Probe Library for \(\text{EG} f_1\)

The logical operator \(\text{EG}\) means that there is a path so that all future states on it are satisfied. Based on this property, we set a rule for the probe library \(Y\) of \(\text{EG} f_1\) as follows.

**Rule 1:** Let \(x_{\text{adv}}\) and \(x_{\text{fir}}\) be two data in \(X\). Then there exists a probe \(x_{\text{w01}}, x_{\text{f1}}\) between them if and only if \(l = t, v = i\) and \(f_1 \in L(v_i), f_1 \in L(v_i)\).

As shown in Fig. 1(d), \(f_1 \in L(v_0), f_1 \in L(v_1)\) and the data \(x_{\text{f1}}\) is adjacent to another data \(x_{\text{w01}}\), so there exists a probe \(x_{\text{w01}}, x_{\text{f1}}\) to connect the two data fibers, which forming a two-aggregation data \(x_{\text{w012}}\) and remaining two data fiber \(x_{\text{w01}}\) and \(x_{\text{f1}}\) [see Fig. 1(e)].

By taking the graph in Fig. 2 as an example, the following steps describe the process of checking \(M, s_0 \models \text{EG} \omega \rightarrow \text{Heat}\) by our approach.

**Step 1:** Data library \(X\) can be constructed as follows:

\[
X = E^2(v_0) \cup E^2(v_1) \cup E^2(v_2) \cup E^2(v_3) \cup E^2(v_4).
\]

where, \(E^2(v_0) = \{x_{01}, E^2(v_1) = \{x_{012}, x_{212}, x_{312}\}, E^2(v_2) = \{x_{121}, x_{123}, x_{124}, x_{221}, x_{243}, x_{242}, x_{423}, x_{424}\}, E^2(v_3) = \{x_{231}\}, and \ E^2(v_4) = \{x_{242}, x_{244}, x_{442}, x_{444}\}\). Let \(\zeta(x)\) represent data fibers of data \(x\). There are total 15 types of data in \(X\), and 30 types of data fibers has been generated as follows:
Based on the data library $X$, we construct the corresponding probe library $Y$ according to rule 1.

Step 2: Probe library construction

$Y_{01} = \{x_{w01}^1, x_{w01}^0\}$
$Y_{12} = \{x_{w012}^2, x_{121}^2, x_{123}^2, x_{122}^2, x_{124}^2\}$
$Y_{21} = \{x_{121}^3, x_{123}^3, x_{122}^3\}
Y_{23} = \{x_{123}^3, x_{232}^3\}
Y_{31} = \{x_{231}^3, x_{321}^3\}$
$Y_{24} = Y_{42} = \emptyset$

Step 3: Probe operations

In each iteration, probes connect the probeable data to form larger data, namely, aggregations. After several rounds, each data containing the initial state is a feasible path satisfying the property, otherwise, the largest data is a counterexample.

Table I: Solutions Procedures

<table>
<thead>
<tr>
<th>Computing Platform</th>
<th>$x_{w01}, x_{012}, x_{212}, x_{312} \ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$x_{w01}[1], x_{012}[3], x_{121}[2], x_{123}[1], x_{232}[1] \ldots$</td>
</tr>
<tr>
<td>1</td>
<td>$x_{w012}[1], x_{012}[3], x_{w012}[3], x_{432}[1] \ldots$</td>
</tr>
<tr>
<td>2</td>
<td>$x_{w012}[1], x_{012}[1], x_{w012}[3], x_{432}[1] \ldots$</td>
</tr>
<tr>
<td>3</td>
<td>$x_{w012}[1]$</td>
</tr>
</tbody>
</table>

In the first iteration, the probe $x_{w01}^1, x_{012}^0$ find the data fiber $x_{w01}^1$ of the data $x_{w01}$ and $x_{012}^0$ of $x_{012}$, connecting them and forming a two-aggregation data $x_{w012}[1]$. The data $x_{124}[4]$ isn’t formed for lack of probe $x_{124}[4]$. Many two-aggregations are generated in this round, but some are not shown in the table.

In the second iteration, some four-aggregations and three-aggregations are generated. For example, the probe $x_{w012}^2, x_{123}^2$ find the data fiber $x_{w012}^2$ of the data $x_{w012}[2]$ and $x_{123}$ of $x_{123}[1]$, connecting them and forming a four-aggregation data $x_{w012}[2]$. In the third iteration, no larger aggregation is produced, so the iteration stops.

It is clear that $x_{w012}[3]$ contains the initial node $s_0$ and a cycle $\pi = s_1 s_3 s_1 s_2 s_3 \ldots$, so it can be concluded that $\mathcal{M}, s_0 \models \mathrm{EG} \neg \text{Heat}$ and an available path is $\pi = s_0 s_1 s_3 s_1 s_2 s_3 \ldots$.

For the problem of $\mathrm{EG}f_1$, all true solutions are the aggregations including the initial state $s_0$ and a cycle, and feasible paths are recorded by aggregations themselves. Otherwise, it means $\mathcal{M}, s_0 \not\models f_1$. Furthermore, the largest aggregation can be the counterexample, since it represents the path closest to true solutions.

Section III-A.
The time complexity of data library $X$ is at most $O(V \times E^2)$.

For each vertex, outgoing and incoming edges need to be visited. Thus, the time complexity of constructing the data library $X$ is $O(V \times E^2)$.

Theorem 2: The time complexity of probe library $Y$ is at most $O(V \times E)$.

We construct the probe library $Y$ by traversing every vertex with its predecessor nodes and its time complexity is $O(e_1)$, and $e_1$ is the number of its predecessor. The total time complexity of this process is at most $O(V \times E)$.

For model checking, the number of iterations is at most equal to $\lceil \log_2 n \rceil + 1$, $n$ is the number of vertexes in the graph $G$. Thanks to the underlying parallelism of the probe machine, the processing ability of one probe operation $\tau$ is $2^q$, $q$ is the number of all possible edges to probe [9]. Therefore, our approach of CTL model checking based on the probe machine can greatly reduce the verification time for systems with large state space.

IV. Prototype Tool

This section is concerned with the framework of our tool CTL2PROBE and simulation of CTL model checking on probe machine. First, we will introduce a model checker called CTL2PROBE based on our approach. And then, some simulations are conducted, which prove the feasibility and efficiency of our approach.

A. CTL2PROBE

To simulate automatic verification of the CTL model checking on the probe machine, we develop a model checker called CTL2PROBE. Fig. 3 shows the framework of CTL2PROBE. It consists of three functional modules: Parsing, Modeling, and Computing. The specific design is as follows:

1. Parsing: This module mainly parses a JSON file into a model and CTL formula. In this process, the transitions between states are recorded in a HashMap structure and the CTL formula is transformed into a parse tree. It takes a particular format JSON file as input. We define this special format to effectively simplify the parsing process. It is a structured markup language similar to XML that users can quickly obtain or change the contents of elements. As shown in Fig. 4, it is a JSON file that describing the state $s_0$ of oven example at Section III. The file includes the label, predecessor, and successor of each state and provides the CTL formula for verification. In the module, CTL formula will be transformed into the formula in terms of $\neg$, $\lor$, $\exists x$, $\forall x$, $\text{EU}$, and $\text{EG}$, and stored in the data structure based on parse tree [see Fig. 5]. It is a bottom-up solving process and the given formula is satisfied when the top node contains the initial node.

2. Mapping: This module is responsible for mapping elements into the data library and probe library. The rule of mapping is the described in Section III. Data library and probe library are provided for the computing module to verify CTL property.

3. Computing: This module is used to implement probe operations according to the data library and probe library. The rule of computing is based on three functions of the computing platform: high cohesiveness, threshold, and uniqueness. We realize them according to the following rules:

   a. High cohesiveness: High-order aggregation data are given higher priority than low-order aggregation to be executed probe operation.

   b. Threshold: The size of two data for probe operation must not exceed the size of the graph itself.

Theorem 1: The time complexity of data library $X$ is at most $O(V \times E^2)$.

The time complexity of probe library $Y$ is at most $O(V \times E)$.

This section is concerned with the framework of our tool CTL2PROBE and simulation of CTL model checking on probe machine. First, we will introduce a model checker called CTL2PROBE based on our approach. And then, some simulations are conducted, which prove the feasibility and efficiency of our approach.

A. CTL2PROBE

To simulate automatic verification of the CTL model checking on the probe machine, we develop a model checker called CTL2PROBE. Fig. 3 shows the framework of CTL2PROBE. It consists of three functional modules: Parsing, Modeling, and Computing. The specific design is as follows:

1. Parsing: This module mainly parses a JSON file into a model and CTL formula. In this process, the transitions between states are recorded in a HashMap structure and the CTL formula is transformed into a parse tree. It takes a particular format JSON file as input. We define this special format to effectively simplify the parsing process. It is a structured markup language similar to XML that users can quickly obtain or change the contents of elements. As shown in Fig. 4, it is a JSON file that describing the state $s_0$ of oven example at Section III. The file includes the label, predecessor, and successor of each state and provides the CTL formula for verification. In the module, CTL formula will be transformed into the formula in terms of $\neg$, $\lor$, $\exists x$, $\forall x$, $\text{EU}$, and $\text{EG}$, and stored in the data structure based on parse tree [see Fig. 5]. It is a bottom-up solving process and the given formula is satisfied when the top node contains the initial node.

2. Mapping: This module is responsible for mapping elements into the data library and probe library. The rule of mapping is the described in Section III. Data library and probe library are provided for the computing module to verify CTL property.

3. Computing: This module is used to implement probe operations according to the data library and probe library. The rule of computing is based on three functions of the computing platform: high cohesiveness, threshold, and uniqueness. We realize them according to the following rules:

   a. High cohesiveness: High-order aggregation data are given higher priority than low-order aggregation to be executed probe operation.

   b. Threshold: The size of two data for probe operation must not exceed the size of the graph itself.

The framework of Verification System is shown in Fig. 3. The data structure of parse tree nodes is presented in Fig. 5.
limitation for state-space explosion and reduce the verification time for systems with large state space. We design a mapping algorithm to transform the model of Kripke structure into the data library and probe library that can run directly on the probe machine. We develop a model checker called CTL2PROBE to simulate the probe machine, which takes the model as input and obtains all feasible paths or counterexamples. Compared to NuSMV, our approach is more efficient by several comparison experiments.

VI. ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development under Project 2019YFA0706404, in part by the NSFC under Project 61972150, and in part by the Shanghai Knowledge Service Platform under Project ZF1213.

REFERENCES


How MOOC Videos Affect Dropout? A Lightweight Pipeline Making Student Dropout Interpretable From Several Levels

Deming Sheng, Jingling Yuan*, Xin Zhang
School of Computer Science and Technology,
Wuhan University of Technology,
Wuhan 430070, China
Email: shengdeming@whut.edu.cn, yjl@whut.edu.cn, xinz@whut.edu.cn

Abstract—Massive Online Open Courses (MOOC) have popularized educational opportunities for students all over the world, while immensely high dropout is becoming a central challenge nowadays. Most researches predict course dropout labels through analyzing the student engagement data. However, these models have high structural complexity with high time cost and cannot provide in-depth insights into why a student is likely to drop out. We devise a lightweight pipeline to simplify the MOOC dropout problem, grasp the core features to make student behaviours interpretable at the model and instance level, visualize the changing trend of predicted label probability estimation with feature values for longitudinally interpreting the sample student behaviours. Based on qualitative insights and quantitative analysis, our main findings are that shorter videos and instructors speak fast are more engaging. Most students complete MOOC learning with a rapid speed, while a few students who watch the video slowly have a higher completion rate. When the frequency of fast-forwarding increases while the percentage of videos watching decreases, the likelihood to drop this course raises. In the end, our pipeline achieves 69.52% AUC, 0.744 $R^2$-squared and 0.553 $\bar{R}^2$-squared with 0.982s inference time on the 20238 sample student data.

Index Terms—MOOC, Dropout, Student Behaviours, Lightweight Pipeline, Interpretable Model

I. INTRODUCTION

Massive Online Open Courses (MOOCs), such as Coursera, edX, Udacity and XuetangX, have developed rapidly in recent years and attracted wide attention of both educators and the public all over the world. As of the end of 2019, more than 900 universities around the world have provided about 11,400 MOOC courses, and in 2018 alone, there were 2,000 new courses. However, the rapid growth of the number of courses has resulted in fewer and fewer students for each course. What’s more, research shows that the average course completion rate on edX is only 5% [1], [2], and 4.5% on XuetangX similarly [3]. How to effectively improve the MOOC completion rate of students has become a prominent problem.

Most of the students engaging MOOCs do not complete the courses and drop them out halfway, which hinders the further development of MOOCs. With the student past engagement time, many works attempt to predict whether a student will drop out or not. However, these models have high structure complexity, strict training conditions and high time cost, which inappropriate for the dynamic of MOOC courses and the diversity of attached student watching video behaviours. More importantly, they do not provide in-depth insights into why a student is likely to drop out, which can not offer analysis for instructors to refine the future versions of MOOC.

Though MOOC dropout models are progressively proposed in recent years, still there are many issues to be addressed when it comes to the real scenarios. For instance, the engagement time can not reflect whether a student is actively paying attention to this video or just playing it in the background while multitasking. A comprehensive collection and analysis of student information are costly, which also increases the burden on instructors. Besides, a wide range of features does improve the accuracy of model predictions while impairing interpretability.

We attempt to filter the original video data and extract core features, utilize a simple pipeline to explain the behaviours of student groups withdrawing from courses, and dynamically analyze the actions of individual students. Through this pipeline, instructors can intuitively predict whether a student will drop out this course from a small amount of data, and take countermeasures to help students complete the course. At the same time, instructors can also make targeted changes to the curriculum for the next semester based on historical data. In summary, our main contributions in this paper are three folds:

- We simplify the MOOC dropout problem and describe some preliminary to make the pipeline interpretable. After that, we explore data analysis, feature engineering, model selection, model metrics
and visualization as design parameters in the context of our lightweight pipeline. In the end, our pipeline achieves 69.52% AUC, 0.744 R-squared and 0.553 R-squared with 0.982s inference time on the 20238 sample student data.

- We adopt two interpretable models to provide in-depth insights into students dropout at several levels. Besides, we visualize the probability estimation and change trend of predicted label linked to feature values to longitudinally interpret the sample student behaviours.
- We devise a deterministic finite automaton to construct the complete student behaviours, which overcomes the inherent limitation of the student engagement time. Moreover, we find that shorter videos and instructors speak fast is more engaging, which enables instructors to adjust MOOC videos for future education.

II. RELATED WORK

Many recent works mainly employ the clickstream as the student engagement and attempt to predict dropout thanks to the detailed records of the students’ interactions with course content, including video lectures, discussion forums, assignments, and additional course content, within the MOOC platform. Kloft et al. [4] propose a Support Vector Machine (SVM) framework focusing on clickstream data, which takes the weekly history of student data into account. Liu et al. [5] employ K-means to make a quantitative analysis of the low completion in MOOCs. Lisas, courses, videos, behaviours, labels \( \mathcal{C}, \mathcal{V}, \mathcal{B}, \mathcal{L} \) set of students, courses, videos, behaviours, labels \( F_{\text{wt}}, F_{\text{wp}} \) feature set of the number and percentage of students watch videos \( F_{\text{st}}, F_{\text{stP}}, F_{\text{st}} \) feature set of the time to students watch the video, progress the video, stay on the webpage \( F_{\text{f}}, F_{\text{fp}}, F_{\text{f}} \) feature set of the frequency of students fast forward, slow backwards, pause, leave the videos \( F_{\text{fl}}, F_{\text{fp}}, F_{\text{fr}} \) feature set of the subtitle length, duration time, speed of videos \( \varepsilon_{\text{f}}, \varepsilon_{\text{p}}, \varepsilon_{\text{f}} \) the upper limit parameter of the student's fast forward, slow backwards, pause, and leave behavior \( \eta_{\text{f}}, \eta_{\text{p}}, \eta_{\text{f}} \) the lower limit parameter of the student's fast forward, slow backwards, pause, and leave behavior

III. PRELIMINARY

Preliminary Before proposing our methodology, we describe some preliminary of our pipeline and corresponding notations (Table I). Given a specified student-course pair \( s \in \mathcal{S}, c \in \mathcal{C} \), we attempt to make the dropout result \( l \in \mathcal{L} \) interpretable after a series of the behaviours \( \{b_1, b_2, ..., b_n\} \subseteq \mathcal{B} \) about watching these MOOC videos \( \{v_1, v_2, ..., v_n\} \subseteq \mathcal{V} \).

Definition 1. Dropout: We define dropout in this paper as meaning that the student will not continue to study the remaining videos of the course after the behaviour of a certain video (not the final video) of a certain course ends.

Definition 2. Student Behaviours: We define student behaviours \( \mathcal{B} \) as the behaviours of each student \( s \) watching a series of videos \( \mathcal{V} \) in the course \( c \). We adopt the tuple \( (s, v, F_{\text{wt}}, F_{\text{wp}}) \) to infer whether the student \( s \) has the action of fast forward or slow backwards and the tuple \( (s, v, F_{\text{st}}, F_{\text{stP}}, F_{\text{st}}) \) to infer whether the student \( s \) has the action of pause or leave in this video \( v \).

Definition 3. Interpretable: We make efforts to explain the final predictions of student dropout models for advancing the development of MOOC. In this paper, interpretability is an explanation, usually in a way that humans can understand, associating the feature values of an instance with its model predictions.

IV. THE PROPOSED LIGHTWEIGHT PIPELINE

A. Overview of the pipeline

As shown in Fig.1, our lightweight pipeline mainly consists of five components. We adopt this pipeline to obtain quantitative results and qualitative insights about the impact of MOOC videos on student dropout and utilize visualization for instructors to intuitively judge the impact of each student behaviour on the final label. We can infer from Fig.1 that Logistic Regression and Decision Tree can provide a fast and accurate prediction on the student dropout rate. And both of them have an excellent model level interpretation to uncover the underlying factors of student withdrawal.
B. Data Processing and Feature Engineering

Data Insights Our dataset comes from the competition “MOOCube student behaviour analysis”, and data was crawled from Chinese one of the largest MOOC platform named XuetangX. XuetangX has provided over 1,000 courses and attracted more than 10,000,000 registered students, which offers abundant data to analyse students behaviours.

Basic Features This competition provides three types of JSON files about courses, videos and student behaviours. A series of basic information can be obtained from these original files, but some are not intuitive enough to make students behaviours interpretable, and some are inappropriate. We force on analyzing the behaviours of each student watching these videos because it is a necessary (but not sufficient) premise for learning and can be quantified by calculating the different times of each student watching these videos. Moreover, the characteristics of the course (video) itself are quite important for students’ MOOC completion rate.

Student Behaviours Construction - The left figure illustrates the construction of complete student behaviours based on deterministic finite automaton (DFA) and the right equation shows the calculation process of these corresponding features.

Student Behaviours Features Video interaction data in XuetangX is recorded for each student generated event separately. These data are recorded in the form of JSON type events, which we then structure into students behaviours features form as described below. Every row in the raw JSON file records a series of times like local_watching_time, video_progress_time, video_start_time and local_start_time. We attempt to aggregate these into simple, interpretable features for a specified video id-student id pair: watching_count, local_watch_time, local_watch_percentage, local_progress_time, local_web_time, forward, backward, pause and leave. Each of these captures various potential factors in the consumption of video content by students, which is shown in Table II.

We employ a deterministic finite automaton (DFA) to construct the complete student behaviours. Fig.2 shows three state transitions in this DFA: Ready, Record, Judge. When the state is Ready, it stays until receives a Forward or Backward event (Pause or Leave, \( F_{\text{sf}} = 1 \) or \( F_{\text{sb}} = 1 \) & \( F_{\text{sp}} = 1 \) or \( F_{\text{sl}} = 1 \)), then the state transforms to Record. At the Record state, there is a stack. When getting a new Pause or Leave (Forward or Backward) events, it pushes all the events into the stack. If there come some other events, it goes to the Judge state. At the Judge state, we check whether the Action.json is normal data to decide whether to save it and return to the Ready state. It is worth noting that in each cycle, \( F_{\text{sf}} \) and \( F_{\text{sb}} \) (\( F_{\text{sp}} \) and \( F_{\text{sl}} \)) can exist at the same time.

Course (Video) Features Each course has a list of corresponding videos, and we utilize the characteristics of these videos to represent different courses. For the original video JSON file, it is difficult to directly use the text information. We make efforts to classify these videos by the video name, but because of the limitation of information, judging based on only a few characters is time-consuming and meaningless. Here we adopt a more direct and effective strategy. We care less about the content of video text itself but pay more attention to the coarse-grained features such as the duration of videos and the speed of speech. The text of all frame is counted and combined with the duration time to calculate the speed of every video. Intuitively, videos where instructors speak fairly fast and with high enthusiasm are more engaging, which is shown in Table II.

Linear Correlation We utilize the Pearson Correlation Coefficients (PCCs) to explore the linear correlation between different features and student dropout (Eq.1).
\[ \rho_{f_1, f_2} = \frac{\text{cov}(f_1, f_2)}{\sigma_{f_1} \sigma_{f_2}} = \frac{E((f_1 - \mu_{f_1})(f_2 - \mu_{f_2}))}{\sigma_{f_1} \sigma_{f_2}} \] (1)

Where \( \text{cov} \) and \( \sigma \) are covariances and standard deviations of two different continuous feature variables, respectively. Each student will produce a series of actions, so the feature parameters are a list. In order to simplify the parameters, we explore the linear correlation between the maximum, minimum, median and average value of each feature and the dropout. Take feature \( F_{wp} \) as an example, \( \rho_{\text{max}(F_{wp}), L} = -0.2825 \), \( \rho_{\text{min}(F_{wp}), L} = 0.0273 \), \( \rho_{\text{avg}(F_{wp}), L} = -0.0193 \), here we will select the feature value with the largest linear correlation as our final input, that is, when the absolute value of \( \rho \) is the largest.

Table II verifies our above conjecture, the completion rate of MOOC course with more video content and instructors speaking fast is higher [Eq.2]. In the above, we introduce the “odds” represents the probability of event divided by the probability of no event (Eq.3).

\[ P(l^i = 1) = \frac{1}{1 + e^{-\beta_0 + \beta_1 f_1 + \beta_2 f_2 + \ldots + \beta_n f_n}} \] (2)

\[ \text{odds} = \frac{P(l^i = 1)}{1 - P(l^i = 1)} = e^{\beta_0 + \beta_1 f_1 + \beta_2 f_2 + \ldots + \beta_n f_n} \] (3)

\[ \text{odds}(f_n + \Delta f) - \beta_n f_n = e^{\beta_n \Delta f} \] (4)

At last, An increase in a feature \( f_n \) by \( \Delta f \) changes the odds ratio (multiplicative) by a factor of \( e^{\beta_n \Delta f} \). Although interpreting the advantage ratio requires some mental arithmetic, it is much simpler than considering the log function. Particularly, for the numerical feature, if the value of feature \( f_n \) increases by one unit, the estimated odds change by a factor of \( e^{\beta_n} \). And for the binary categorical feature, altering the feature \( f_n \) from the reference category to the other category changes the estimated odds by a factor of \( e^{\beta_n} \) as well.

**Model Level Interpretation** In the above, we introduce odds to more intuitively show the impact of each feature change on the prediction results, for the interpretation of weight in the logistic regression model depends on the type of the corresponding feature. Another important measurement for interpreting models is the \( R^2 \)-squared measurement, which implies the total variance of the target result is explained by the model (Eq.5).

\[ R^2 = 1 - \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\sum_{i=1}^{n} (l^i - \bar{l})^2}{\sum_{i=1}^{n} (l^i - \bar{L})^2} \] (5)

Where SSE is the squared sum of the error terms, which is measured by the squared differences between the predicted and actual target values. And SST is the squared sum of the data variance, which is measured by the square difference between the average and actual target value. \( R^2 \)-squared increases with the number of features in the model, but it is not related to the number of instances and labels. Here, we also list the adjusted \( R^2 \)-squared (\( \bar{R} \)-squared) values to account for the number of features used in the model. The calculation method of \( \bar{R} \)-squared is shown in Eq.6, where \( i \) is the number of features and \( n \) the number of instances.

\[ \bar{R}^2 = 1 - (1 - R^2) \frac{n - 1}{n - i - 1} \] (6)

As we can see from Table III, the values of \( R \) and \( \bar{R} \) are both increasing with the number of input features, which indicates that the interpretability of the logistic regression model will strengthen with the richness of input features.
TABLE III
MODEL PERFORMANCE - THE INFLUENCE OF DIFFERENT FEATURES ON THE INTERPRETABILITY AND ACCURACY. WE ADD THEM ONE BY ONE ACCORDING TO THE IMPORTANCE OF THE FEATURE (REFER TO THE ESTIMATE AND SIGNIF. CODES COLUMNS OF TABLE IV FOR THE RELEVANT IMPORTANCE).

<table>
<thead>
<tr>
<th>Model (L = ∑ (F_i))</th>
<th>AIC</th>
<th>BIC</th>
<th>(R^2)</th>
<th>(R^2_{\text{adj}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F_{wp})</td>
<td>23780</td>
<td>23793</td>
<td>0.985642</td>
<td>0.4879666</td>
</tr>
<tr>
<td>(F_{wp} + F_s)</td>
<td>22572</td>
<td>22569</td>
<td>0.9956793</td>
<td>0.4979337</td>
</tr>
<tr>
<td>(F_{wp} + F_s + F_b)</td>
<td>29004</td>
<td>29063</td>
<td>0.7071946</td>
<td>0.5000501</td>
</tr>
<tr>
<td>(F_{wp} + F_s + F_b + F_v)</td>
<td>20451</td>
<td>20491</td>
<td>0.7246000</td>
<td>0.5203239</td>
</tr>
<tr>
<td>(F_{wp} + F_s + F_b + F_v + F_t)</td>
<td>20447</td>
<td>20494</td>
<td>0.7262697</td>
<td>0.5220901</td>
</tr>
<tr>
<td>(F_{wp} + F_s + F_b + F_v + F_t + F_p)</td>
<td>20445</td>
<td>20500</td>
<td>0.7271775</td>
<td>0.5286474</td>
</tr>
</tbody>
</table>

However, this does not mean using all variables, because too many variables will lead to over-fitting, which will reduce the generalization of the model. In addition, complex features are not suitable for model interpretability. In this paper, we hope to find a small number of decision-making features to help MOOC instructors infer whether students will complete this course for future course adjustments.

We simplify the input to improve the interpretability of the model by eliminating linear irrelevant and collinearity features. Here we employ two indicators, Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC), to verify that the 8 modelling variable models we selected alleviate the problem between accuracy and overfitting.

\[
AIC = -2 \ln(L) + 2i, \quad BIC = -2 \ln(L) + i \ln(n) \tag{7}
\]

Where \(L\) is the maximum likelihood under the logistic regression model, \(n\) is the number of data, and \(i\) is the number of input features in this model. Table IV shows that the model we built balances the complexity of the model and the ability to describe the data set (likelihood function).

It can be seen from Table III and Table IV that the course completion rate is the most relevant to the behaviours of student watching videos, forward (\(F_s\)) and backward (\(F_b\)). What’s more, the dropout rate of courses with slower video speaking rates will increase. In general, the logistic regression model gives more weight to preprocessed features. Taking video watching time (\(F_{wt}\)) and video watching rate (\(F_{wp}\)) as examples, both of them can measure the degree to which a student learns a certain video. But from the actual data, we can see that \(F_{wp}\) is more sensitive, which means it is essential to extract student behaviours from basic information.

**Instance Level Interpretation** We visualize the co-efficients of the logistic regression model to explain the result of the sample student’s final label after a series of actions (Fig.4(b)). For individuals, basic features and student behaviours features have a linear and negative correlation with the MOOC dropout. That is to say, the higher the time or the percentage of an example student watching the video, the more frequently playing forward or backwards, the higher likelihood he is to complete this course. Video features are positively linearly correlated with the MOOC dropout, the speed of the video also affects other features. As shown in Fig.4(c), when the influence of \(F_{wp}\) increases, the influence of other features will decrease. At this time, the student is more likely to give up the course.

**D. Decision Tree Model**

**Decision Tree** Decision tree models can handle situations where the relationship between features and outcome is nonlinear or where features interact with each other. Decision tree models split the data multiple times according to certain cutoff values in the features. Through splitting, different subsets of the dataset are created, with each instance belonging to one subset. The final subsets are called terminal (leaf) nodes and the intermediate subsets are called internal (split) nodes. The following equation describes the relationship between the outcome label \(L\) and the input features \(F\).

\[
M = \sum_{m=1}^{M} c_m I \{ f \in T_m \} \tag{8}
\]

Each instance falls into one terminal node subset \(T_m\). \(I \{ f \in T_m \}\) denotes the identity function which return 1 when the input features \(F\) is in the final subsets \(T_m\), 0 otherwise. For a terminal node \(l_i\), the instance outcome label is \(l_i = \bar{c}\), where \(\bar{c}\) denotes the average value of the whole instances in this terminal node subset \(T_m\).

**Model Level Interpretation** We finally select the five most relevant features to construct the decision tree model through the previous analysis. As the max_depth of the decision tree model increases, the accuracy of the model will be improved, and the corresponding feature weights will be more balanced. But it is worth noting that the
Table VI

<table>
<thead>
<tr>
<th>Model Instance</th>
<th>Drop or Complete Instances under Different Depth Decision Tree Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>max_depth=4</td>
<td>Instance</td>
</tr>
<tr>
<td>Drop</td>
<td>$F_s \geq 100, F_p \geq 240, 000$</td>
</tr>
<tr>
<td></td>
<td>0.023</td>
</tr>
<tr>
<td></td>
<td>2106</td>
</tr>
<tr>
<td>max_depth=5</td>
<td>$F_s \geq 90, 875, F_p \geq 324, 007$</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>72</td>
</tr>
<tr>
<td>max_depth=6</td>
<td>$F_p \geq 141, 546, F_s \geq 245, 441, F_s \geq 9.5$</td>
</tr>
<tr>
<td></td>
<td>0.069</td>
</tr>
<tr>
<td></td>
<td>1127</td>
</tr>
<tr>
<td>max_depth=7</td>
<td>$F_s \geq 122, 564, F_p \geq 9, 860, F_p \geq 300, 331$</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>188</td>
</tr>
<tr>
<td>Complete</td>
<td>Instance</td>
</tr>
<tr>
<td>max_depth=4</td>
<td>$F_s \geq 59, 075, F_p \geq 240, 247$</td>
</tr>
<tr>
<td></td>
<td>0.225</td>
</tr>
<tr>
<td></td>
<td>114</td>
</tr>
<tr>
<td>max_depth=5</td>
<td>$F_s \geq 56, 263, F_p \geq 304, 025$</td>
</tr>
<tr>
<td></td>
<td>0.03</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>max_depth=6</td>
<td>$F_s \geq 56, 872, F_p \geq 150, 319, F_p \geq 4.5$</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>18</td>
</tr>
<tr>
<td>max_depth=7</td>
<td>$F_s \geq 43, 267, F_p \geq 240, 725$</td>
</tr>
<tr>
<td></td>
<td>0.04</td>
</tr>
<tr>
<td></td>
<td>48</td>
</tr>
<tr>
<td>max_depth=8</td>
<td>$F_s \geq 53, 106, F_p \geq 9, 867, F_p \geq 144, 699$</td>
</tr>
<tr>
<td></td>
<td>0.08</td>
</tr>
<tr>
<td></td>
<td>87</td>
</tr>
</tbody>
</table>

Fig. 4. Instance Interpretation - The features of the dropout prediction model have different priorities. As students’ actions of watching videos increase, these features will change differently (the feature change interval is shown in subfigure (a)), and the estimated weight of the feature will also change (the trend of feature change is shown in subfigure (b)).

V. Conclusion

Our lightweight pipeline begins with interpretable features, we employ a deterministic finite automatons to construct the complete student behaviours and we adopt a direct and effective strategy to extract the course features. We do our utmost to simplify the MOOC dropout problem, grasp the core features through two interpretable models. In the end, the student’s dropout behaviour is explained at several levels, which enables instructors and video producers to make the most of online videos for future education.

References


Development of an Automated Machine Learning Solution Integrable With Multiple Virtual Learning Environments

Raniel Gomes da Silva¹, Vitória Maria Pena Mendes¹, Rodrigo Lins Rodrigues², and Alexandre Magno Andrade Maciel¹

¹Universidade de Pernambuco
²Universidade Federal Rural de Pernambuco

Abstract

In the last decade, a large volume of data has emerged from the massive use of Virtual Learning Environments (VLE). The information contained in these data has enabled the evolution of Educational Data Mining (EDM), whose objective is to apply Machine Learning (ML) in educational contexts. However, building accurate and robust ML models requires, in most cases, advanced knowledge in data science. To solve such problems, Automated Machine Learning techniques have been studied, to simplify the repetitive processes of Data Mining. To validate the solution, the database of the Núcleo de Educação a Distância da Universidade de Pernambuco was used. In comparison with the classic EDM approaches, the applied technique showed a superior result, obtaining an accuracy of 89% in the student performance classification process. This solution is called the Framework de Mineracão de Dados Educativos (FMDEV), whose objective is to allow users to validate and make available ML baselines with greater productivity. The results of the experts’ opinions prove that the FMDEV can contribute to the construction of better models of ML.

1 Introduction

In the last decade, the adoption of distance learning tools has grown exponentially. Consequently, a large volume of data has emerged from the massive use of Virtual Learning Environments (VLE) [1]. For knowledge extraction from these data, it is necessary to carry out a series of data mining processes [2].

Educational Data Mining (EDM) techniques have been adopted frequently, as an alternative for extracting knowledge from data obtained from VLE [3]. The EDM process is conceptualized as a paradigm for building models, tasks, methods, and algorithms from educational databases. Within this context, the use of supervised learning is quite common to solve tasks such as analyzing student performance or predicting evasion risk. [4].

Machine Learning (ML) application is important for data scientists, tutors, and teachers. However, given the complexity of educational problems, the model building requires advanced knowledge in data science [5]. In addition, other factors such as the development time in model building, the definition of input parameters, and the selection of the best algorithm, make it impossible for the use of ML to be democratized for non-technical users. [6].

To solve such problems, Automated Machine Learning (AutoML) techniques have been studied, with the objective of simplifying repetitive Data Mining processes, which do not require domain knowledge in most cases [7]. Bayesian Optimization (BO) and Evolutionary Algorithm (EA) techniques have been applied in the categories of Automated Engineering of Features (AutoFE) and in the Automation of Models with Hyperparameter Learning (AutoMHL) [8].

Some technologies simplify such steps, such as TPOT [9, 10], AutoKeras [11] and Auto-Weka [12], however, it is necessary to have a minimum of knowledge in data science [13]. Tools like the one by Fusijawa et al., present a closer path for a non-technical user, however, the developed features require knowledge in Structured Query Language (SQL) and also ML, as the user needs to define an algorithm for training [14].

Thus, the justification for this work lies in the need to develop an AutoML solution that favors users in the extraction of knowledge from EDM, even with little experience in ML techniques. Considering the various possibilities of applications of this solution, this research focused on EDM, with the integrability of multiple VLE, to obtain evidence in the context of a course, discipline, period, or class.

The general objective of this article is to develop an Automated Machine Learning solution that can be integrated into Virtual Learning Environments or data visualization tools, based on the application of models generated through Genetic Algorithm techniques. To achieve the general objective, the following specific objectives were established:
Identify the essential functionalities of an Automatic Machine Learning solution for use in Educational Data Mining; implement an application with the proposed solution and evaluate the implemented solution from experiments with the database of the Núcleo de Educação a Distância da Universidade de Pernambuco (NEAD) and measure the quality of the framework from integration tests.

2 Background and Related Work

2.1 Machine Learning

Machine Learning has as its definition the application of computational methods that obtain expertise, with the objective of improving performance or applying partial predictions in a given context [15]. In practice, prediction algorithms are built with high robustness, which depends on a sample necessary for the algorithm to learn a family of concepts [15]. Contextualizing this task for the CRISP-DM, it is possible to fit the ML in the pre-processing, modeling, and evaluation steps.

As for the types of ML algorithms, there are the supervised learning, unsupervised learning, semi-supervised and reinforcement learning [16]. The definition of the correct type for each context may depend on some criteria, such as (I) The need for human supervision as to the expected output; (II) Speed of relearning as new input data is available; (III) The identification of new patterns based on trained characteristics, or simply, classifying them considering a data entry [16].

2.2 Automated Machine Learning

Automated Machine Learning (AutoML) is an abstraction for ML in which it proposes to optimize productivity in the pre-processing and modeling steps. Steps such as selecting features, defining the best algorithm and configuring hyperparameters, are built automatically and without human intervention [17]. Another value delivered by the AutoML approach is to prevent the data scientist from wasting development time on repetitive trial and error tasks [17].

There are categories in AutoML with a greater focus on the selection of features (AutoFE) and others with a greater emphasis on the definition step of model optimization and hyperparameters (AutoMHL), finally cases more focused on Deep Learning (AutoDL). Among the techniques most applied in AutoML, there are Bayesian Optimization (BO) and Evolutionary Algorithm (EA) [8]. There are also applications based on techniques with Reinforcement Learning and Gradient-based, but they are still very incipient [8].

Figure 1 defines a conventional ML flow. According to the diagram above, the use of AutoML seeks to optimize the two major areas in evidence. From the point of view of features engineering, three steps seek to be optimized. In the data cleaning step, imputation techniques and attribute normalization are applied. When generating features, new metadata is created from existing ones. In the selection of features, dimensionality reduction techniques (VarianceThreshold, SelectKBest), to optimize the database. From the modeling point of view, BO or EA techniques are applied to identify the best algorithm and hyperparameters (Figure 1).

3 Proposed Solution

The proposed solution is called the Framework de Mineração de Dados Educacionais (FMDEV). FMDEV applies the CRISP-DM [18] methodology to the solution steps. On the construction of the solution, Requirements Engineering [19] and Lean Inception [20, 21, 22] techniques were applied to define the necessary functionalities for FMDEV.

FMDEV is able to assist in data pre-processing, training, validation, and availability. The steps of understanding the business and understanding the data are not the responsibility of FMDEV. FMDEV makes the models available in REST format so that data visualization tools and Virtual Learning Environments are able to consume the endpoints generated by FMDEV.

Before the actual implementation process, a navigable prototype was built using the tool Figma [23]. The main stream of Framework de Mineração de Dados Educacionais (FMDEV) is divided into four screens: (I) Data sources; (II) Selection of indicators; (III) Data pre-processing and (IV) Training. For the trained and saved models, a screen was created separately from the main flow.

4 Results

The assessment of the environment proposed in this work sought to validate the FMDEV under four sets of experiments: the first analyzed the use of it as a tool to assist in the generation of educational data mining models; the second assessed the holistic functioning of the environment based on integration tests; the third, carried out an opinion
<table>
<thead>
<tr>
<th>Course</th>
<th>ROC AUC</th>
<th>Recall</th>
<th>Precision</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>96.42%</td>
<td>80.33%</td>
<td>86.80%</td>
<td>83.42%</td>
</tr>
<tr>
<td>Biology</td>
<td>93.34%</td>
<td>80.32%</td>
<td>87.34%</td>
<td>83.67%</td>
</tr>
<tr>
<td>Literature</td>
<td>88.77%</td>
<td>77.39%</td>
<td>82.94%</td>
<td>80.06%</td>
</tr>
<tr>
<td>Pedagogy</td>
<td>93.16%</td>
<td>78.25%</td>
<td>87.58%</td>
<td>82.65%</td>
</tr>
</tbody>
</table>

Table 1: Bayesian Optimization Technique Results.

For each scenario, the base was divided into 70% training and 30% tests. As for the evaluation metrics, ROC curve (AUC), recall, precision, and F1 Score are present. All procedures were performed on a Linux server (Intel Core i7 2.2 GHz; 4 Cores; 16 GB of RAM). According to Table 1, the first round of tests included the BO technique.

AutoKeras library was applied to assist the execution of the AutoML technique for Bayesian Optimization [24]. It is possible to notice that the Administration course obtained the best performance in the tests (89.36%). The Literature course had the lowest accuracy (84.23%). In the 2 table, the results with the GA technique and the due considerations regarding the two techniques will be presented.

<table>
<thead>
<tr>
<th>Course</th>
<th>ROC AUC</th>
<th>Recall</th>
<th>Precision</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>96.62%</td>
<td>84.51%</td>
<td>91.32%</td>
<td>88.04%</td>
</tr>
<tr>
<td>Biology</td>
<td>95.93%</td>
<td>84.37%</td>
<td>90.95%</td>
<td>87.47%</td>
</tr>
<tr>
<td>Literature</td>
<td>94.23%</td>
<td>86.87%</td>
<td>87.16%</td>
<td>87.01%</td>
</tr>
<tr>
<td>Pedagogy</td>
<td>96.11%</td>
<td>84.92%</td>
<td>88.99%</td>
<td>86.90%</td>
</tr>
</tbody>
</table>

Table 2: Genetic Algorithm Technique Results.

TPOT library was applied to assist the execution of the AutoML technique for Genetic Algorithm [25]. Specifically for this technology, it is possible to create a parallelism rule, in which all colors of the experiment machine can be used [26].

As for the models optimized for each experiment with GA, the following algorithms were obtained: XGBoost (Administration and Biology), Random Forest (Literature) and Extra Trees (Pedagogy) [27, 28, 29]. It is interesting to note that the administration and biology courses obtained the same algorithm, however, their hyperparameters, defined from the multiple generations in the AutoML technique, obtained completely different configurations.

When comparing the metrics of GA and BO, it is noticeable that the GA excels in all cases. From an average among all courses, the accuracy of GA is 2.52% higher; AUC at 2.79%, recall at 6.08%, precision at 3.57% and F1 Score at 4.90%. Of the four courses evaluated, the greatest discrepancy in techniques is presented in the Literature course. The F1 Score generated by the GA technique for this course, is 6.94% higher. Given this, it is important to note several contributions with the use of the GA technique in AutoML: superior results in relation to the BO technique and explanability of the models and the optimized hyperparameters.

5 Conclusion

This work proposed the development of an Automated Machine Learning solution for EDM. For this, a framework was developed capable of using Moodle data sources, as well as CSV files that can be imported directly into FMDEV. The framework allows you to create, manage, and consume supervised classification models in a simple way for non-technical users and productive for technical users or with little expertise in data science.

Regarding the machine learning area, contributions were reported from the use of the Genetic Algorithm techniques, in comparison with the Bayesian Optimization technique. For the case study applied with the NEAD database, it was possible to perceive that the results presented from the GA technique, was superior in all scenarios, in comparison with...
the BO technique. In addition, the GA technique showed superior results compared to conventional data mining techniques, considering the case study addressed.

Regarding the software engineering area, integration tests were developed for all available endpoints. Such tests ensured that the FMDEV presents consistency and conformity based on the functional and non-functional requirements presented. In addition, these tests will be useful as a software quality assurance strategy.

As a way of ensuring more reliability in this research, an expert opinion was carried out to assess the conformity of the steps developed in the FMDEV. The experts’ feedback corroborated that FMDEV is able to simplify the process of mining educational data, as well as promoting productivity in the use of automated machine learning. The fact that FMDEV is able to abstract the complexity of Machine Learning algorithms, shorten the development time of models and remove the difficulty in defining the parameters of the algorithms, will enable a great differential in the processes of EDM.

Acknowledgement

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - finance code 001.

References


Towards the integration of the GDPR in the Unified Software Development Process

Elena Gómez-Martínez, Miguel Marroyo, Silvia T. Acuña
Departamento de Ingeniería Informática
Universidad Autónoma de Madrid, Madrid, Spain
{MariaElena.Gomez, Miguel.Marroyo, Silvia.Acunna}@uam.es

Abstract—The General Data Protection Regulation (GDPR) is the core of digital privacy legislation across Europe (EU), and it applies to processing carried out by organisations operating within and outside the EU that offer goods or services to individuals in the EU, including software products. Nevertheless, software teams are generally unaware of the legal requirements for personal data protection and its application throughout the software life cycle. In this paper, we propose a comprehensive guidance to integrate compliance with GDPR requirements within the Unified Software Development Process (UP) across the entire lifetime.

Keywords—data protection regulation; unified software development process; personal data; privacy requirements.

I. INTRODUCTION

As part of the software process activities, software engineers should get acquainted with and understand guidelines related to information privacy, as software requirements should comply with data privacy laws. Therefore, development team members need to be familiar with the applicable personal data protection legislation [12]. In addition, the identification of the privacy requirements, that is, the protection of personal data that are processed by complex systems, is a tough, error-prone task in social networks where users often enter personal and other sensitive data that would otherwise be subject to varying levels of personal data protection, security and privacy [25].

The European Union (EU) General Data Protection Regulation 2016/679 (GDPR) comes into force in May 2018 [13], representing an advance in personal data protection. For enterprises in the EU or those that work with resident data in it, this regulation means a new challenge, namely, to avoid costly fines (up to 20 million or 4% of turnover) and offer their clients a service that guarantees their privacy rights. Not only should software engineers validate that installed systems comply with privacy requirements, personal data and data protection needs should also be identified during the early development activities, including requirements capture and analysis, in order to specify the associated requirements [12][19][9]. For instance, the personal data gathered from users by developers designing a medical data management application, where age or sex are important, will not be the same as for a library management system, where such data are unnecessary. If these data and needs are identified later during the development process, it will be more costly to solve data privacy-related problems, because the changes that have to be made to the future system will tend to affect more functionalities [3][18][22]. Nevertheless, the software development teams do not have a framework for adopting the personal data protection legislation in software development, as the this legislation should change development team work methods by, for example, adopting a series of features and controls related to consent, documentation and privacy responsibilities throughout the software development process [4][5][26].

Regarding the lack of a framework for adopting the personal data protection legislation in software development, in 2017, in a survey conducted by PwC [23], 92% of organizations in the United States of America believe that even though GDPR is a European regulation, it still affects their business and therefore compliance should be a priority. In the same year, Deloitte conducted a survey of organizations in EMEA (Europe, Middle East and Africa) [11], in which only 15% of companies believed that they could fully comply with regulations by May 2018. Companies like Facebook or Google have been fined US$114 million, and countries such as Greece, Portugal or Slovenia have not adjusted their measures to adapt to their national regulations [14]. Therefore, despite the importance of GDPR has been recognized, due to the lack of a defined framework to incorporate GDPR into the software development process, it is difficult to fully take action to comply with the legislation.

The objective of this article is to provide a framework from the point of view of software engineering that incorporates the European General Data Protection Regulation in the Unified Software Development Process or Unified Process (UP), thereby software teams can define and specify the privacy requirements in early development activities and throughout the software development process, including design, implementation, testing, and maintenance.

The research question is if: it is feasible to apply the GDPR into the UP. GDPR [13] has important value in the technical and information technology fields at European and international levels, so GDPR must be considered in any process related to software development. In addition, it guarantees that if you follow its instructions, the processing of personal data will be transparent, honest, and safe, which is very important for companies and organizations as well as users themselves. The Unified Process [17] has been selected thereby that it can be tailored to a wide range of software development projects and provides a formalised prescription of the entire software development process, as it specifies all the software process modelling elements through Unified Modelling Language (UML) [21]. The research method used consists of the analysis and synthesis of the GDPR and its justified inclusion in the UP following the standard set by the Data Management Association (DAMA) International, known for its data management guide [10]. To verify this adaptation, the Regulatory Compliance List developed by the Spanish Agency for Data Protection (in Spanish, Agencia Española de...
The system is already in use.

Conclusions and future work. Finally, Section V outlines the

introduces key basic concepts. Section III presents our approach
quirements elicitation to deployment to the customer and

A development team with the legal requirements of the regula-
achieved following our proposal. All this work has been carried

Garantías de los Derechos Digitales

The rest of this paper is organized as follows. Section II in-

II. BACKGROUND

A. Legal Environment: GDPR and LOPD

The GDPR [13] is the core of Europe’s digital privacy legis-
more than its predecessor the Data Protection Directive or Di-
apply to organisations in all member states

The GDPR provides the following rights for individuals: a) The
right to be informed, Articles 13 and 14 of the GDPR specify
individuals have the right to have their personal data rectified; d) The right
to rectification, under Article 16 each individual has the right to
have inaccurate personal data rectified; d) The right to erasure,
individuals have the right to access their personal data; e) The right
to restrict processing, Article 18 gives individuals the right to restrict the processing of their personal
data in certain circumstances; f) The right to data portability,
individuals have the right to receive personal data they have pro-
vided to a controller in a structured, commonly used and ma-
chine-readable format; g) The right to object, Article 21 gives
individuals the right to object to the processing of their personal
data at any time; h) Rights in relation to automated decision
making and profiling, Article 22 has additional rules to protect
individuals if you are carrying out solely automated decision-
making that has legal or similarly significant effects on them.

The directive entered into force on 5 May 2016 and EU coun-
tries had to transpose it into their national regulations. The Law
on the Protection of Personal Data and Guarantee of Digital
Rights (in Spanish, Ley de Protección de Datos Personales y
Garantías de los Derechos Digitales, in short LOPD) [7] sets out
the data protection framework in Spain, alongside the GDPR.
Despite having some dissimilarities with respect to the GDPR,
they do not impact the software development. Therefore, both
acronyms, LOPD and GDPR, will be used interchangeably in
this paper, employing the latter for the sake of brevity. We focus
on UP process workflows (activities) and all the software develop-

analysing the Spanish LOPD: requirements, analysis, design,
implementation, test and maintenance.

B. Data Management Book of Knowledge

The guidebook named “The DAMA Guide to the Data Man-
agement Body of Knowledge” (DAMA-DMBOK) [10] estab-
lishes a framework to data management standards and practices
for data management, remarking the importance of data quality
and ethics. This framework is structured around the 11
Knowledge Areas with core activities surrounded by software
lifecycle and usage activities, contained within the structures of
governance. Settled within the Knowledge Areas are the essen-
tial objectives and principles of data management. Here we
only focus on those related to the software process.

Data Governance provides the general template and over-
sight to govern data management by establishing a framework
doctrine rights over data that accounts for the company’s
needs, and according to the current legislation. It affects to all
the software lifecycle encompassing from the access policy, us-
age, security and quality to the fulfilment of requirements. Data
Architecture defines the master plan to handle and maintain
data assets by with regards to organizational strategy which are
already establish with other strategic data requirements and de-
signs to meet these necessities. Data Modelling and Design
determines, analyses, represents, and communicates data require-
ments in a detailed form which is called the data model. Data
Security and Operations comprises the design, application, and
maintains of stored data to make the most of its value. Data Se-
curity ensures that data privacy and confidentiality are main-
tained that data is accessed appropriately and not breached
across various channels of use. Data Integration and Interop-

erability includes processes associated with the movement and
consolidation of data within and between data stores, applica-
tions, and organizations. Document and Content Management
are measures and strategies which are used to handle the lifecycle
of data and information in a range of unstructured me-
dia, especially documents needed to support legal, regulatory
compliance requirements and ethics implication. Data Quality
includes the planning and implementation of quality manage-
techniques to measure, assess, and improve the fitness of
data for use within an organization.

III. INTEGRATING THE GDPR

The Spanish LOPD [7] is comprised by 97 articles. Many of
them are transversal to the activities of the software development
process, and therefore they can be affected. Notwithstanding, not
all the articles have a place within any of the activities given its
purely legal nature and that it does not apply to the technological
ccontext. Several articles in the LOPD mention crucial informa-
tion contained in the GDPR. In those cases, we have contem-
plated that information. We will refer as (Art. X) to the Article
labelled X of the LOPD.

A. Procedure

Before analysing the GDPR, we have firstly established a
 correspondence of each Knowledge Area of DAMA-DMBOK
with the UP activities in order to guarantee the data manage-
ment. Note that not all Knowledge Areas have been considered
since they are not included in the UP. This correspondence is summarised in Table I. As it can be observed, data management process and governance are considered globally strategic to the entire software lifecycle. In addition, data security shall affect all the process, otherwise security could be uncompleted, since it is designed, but not implemented. Processes related to architecture, modelling, design, storage and operations belong to design activity of the UP, which considers all the software requirements. Data ethics and management are closely related to the fulfilment of users’ rights; therefore, they correspond to the analysis activity. Since these rights have to be guaranteed during all the software lifecycle, we have included it in the maintenance activity. To integrate the legal requirements imposed by the GDPR into the UP, we propose a procedure with the following steps:

1. Overview of the regulation: Initial reading of every article and extraction of keywords and concepts which can guide us to correlate UP activities within the GDPR.

2. Reading of each article: Each article is analysed in detail by focusing on a set of keywords in Table I and Knowledge Areas of DAMA-DMBOK. After this study, we determined if the article can be applicable to any development activity.

3. In the case the article is applicable, we placed it in one or more development phases, and extracted new keyword if needed and reassessed the examined articles.

4. Otherwise, we evaluated the next article.

The inclusion of the GDPR into the UP has been carried out by an interdisciplinary team composed by senior and junior software engineers together with an expert in data protection and data auditing. For the sake of brevity, additional documents are available at https://github.com/egomez26/SEKE2021.

B. Requirements and Analysis

Since most of the legal requirements are in the initial phases of the project, we have decided to tackle both activities at the same time. All the activities to carry out during this phase are summarised in the workflow in Fig. 1, whose rationale is described below.

<table>
<thead>
<tr>
<th>Elicitation &amp; Analysis</th>
<th>Design</th>
<th>Implementation &amp; Testing</th>
<th>Maintenance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rights and Obligations</td>
<td>Storage</td>
<td>User</td>
<td>Lifetime</td>
</tr>
<tr>
<td>Data type</td>
<td>Transfer</td>
<td>Validation</td>
<td>Data life cycle</td>
</tr>
<tr>
<td>Purpose</td>
<td>Architecture</td>
<td>Interface</td>
<td>Register</td>
</tr>
<tr>
<td>Requirements</td>
<td>Security</td>
<td>Consent</td>
<td>Security</td>
</tr>
<tr>
<td>Processing</td>
<td>Interface</td>
<td>Communication</td>
<td>Control</td>
</tr>
<tr>
<td>Limitations</td>
<td>Communication</td>
<td>Provide information</td>
<td>Rights and Obligations</td>
</tr>
<tr>
<td>Assessment</td>
<td>Measures and codes</td>
<td>Controller Data access</td>
<td>Security</td>
</tr>
</tbody>
</table>

In the early activity of any software development, the data scope is defined, that is, their data type and purpose. According to (Art. 4), data have to be precise and represent the reality. Furthermore, it is mandatory that users express their explicit consent to use their required personal data and the purposes of the processing, and to be informed of this fact (Art. 8). Therefore, we need to identify the minimum information with personal data for our application and to not use this information for any kind of discrimination on the grounds of age, sex, gender and sexual orientation, race or ethnic group, political convictions, health, biometric information, and/or religion (Art. 9). Notwithstanding, there exist exceptions to use the aforementioned data if individual gives her consent or that the data processing is vital in a legal process (Art. 10).

Concerning the purposes of data processing, once the data are collected, the user has the legal right to claim this purpose at any time, requiring the name of the contact detail of the organization, identity of the data protection officer, the lawful basis of the processing, and any related information (Art. 15 to 22), including data categories and their sources (Art. 11). The Spanish LOPD also encompasses the following data processing to protect personality or habits aspects, automated individual decision-making, systematic profiling, monitoring or geolocating of people, genetic information, vulnerable people or at risk of social exclusion, and/or preventing or discouraging from exercising their rights. In all these cases, the application will have to require the user’s acceptance or her legal representative (Art. 12). Obviously, the application has to guarantee all their rights of end-users, and the security tools to achieve it (Art. 80). Therefore, the main functionalities of the software must be considered if this information is processed, e.g., application for banking or video surveillance. We must also define the kind of application, target end-users, processed information, context of use, and derived contents (Art. 87). In the following, we describe the most sensitive applications. In the working environment, since service providers may be a natural person or a corporate entity, information related to self-employed, freelances, individual entrepreneurs and professionals has special attention (Art. 19). In these cases, we will only consider the minimum personal data to locate her professional activity, such as working address and phone number. We must also contemplate the direct or indirect geolocation of workers (Art. 90) or video surveillance at workplaces (Art.
89). If we are developing a banking application, it is lawful to have credit information if that data has been provided by the creditor (Art. 20). It is also lawful when they derived from the development of a commercial transaction, i.e., when the company structure has been modified, acquired or sold (Art. 21). Video surveillance applications can process captured images and videos if the purpose is to guarantee safety of individuals and facilities (Art. 22). Thus, a mechanism to destroy these images or videos in a month from their acquisition must be considered. Marketing communication applications must create an information system to store the indispensable data for those individuals who do not want to receive these communications (Art. 23). Statistical applications can use personal data at the prior disposal of those affected, provided voluntarily (Art. 25). Applications involving underage users, we need to storage the consent of their parents or legal custodians to process their information in the analysis activity (Art. 92). Data controllers, processors and officers play a key role in software applications, since we must contemplate their different responsibilities, role access and functionalities in our system (Art. 28 to 37). A data controller can handle personal data using technical measures to guarantee that the Directive is fulfilled. They are in charge of registering all the processing activities (Art. 28). Data officers are able to lock information (Art. 32). In addition, the application must provide default functionality to erase personal data for individuals who exercise their right to be forgotten (Art. 94), to modify inaccurate or low-quality information (Art. 93), and to port personal data (Art. 95). To guarantee digital security, we will need to analyse it in this phase, considering level of confidentiality for sensitive data, involved roles, administration procedures, maintenance plans, monitoring, auditing and policy compliance (Art. 82), that we will describe in the following sections.

C. Design

Design activities define the data domain and the architecture from those requirements obtained in the previous analysis. One key aspect during design is to guarantee confidentiality in data structures, interfaces and algorithms (Art. 5). With this purpose, the application ensures not authorized data processing, and lost or destruction of personal data. In addition, data must be accurate and, if needed, updated (Art. 4). Designers must take into account all these aspects to prepare databases, including memory requirements, index and precise processing to make queries, generate verified statistics, provide backup services and balance to allow the access at any time. Remark that it is not possible to process data categories into conflict with Art. 9, that is special information, and criminal records (Art. 10). We will also include in databases information such as identification of the data processing officer, if there are data of special categories, and data sources (Art. 11). Software systems must also consider mechanisms to register all the activities related to Art. 31. This register must contain the following fields: name and contact details of the data processing officer, purpose of processing, categories of personal data and person concerned, categories of recipient of those personal data, transaction to third countries or international organizations Art. 82, expected timeframe for removing data, technical and organizational measures to guarantee data security. This register shall have a relation N:N between the Register class and each data category. As a step forward, to facilitate the integration of the legal requirements into design activities, we have also drawn a representation of the GDPR. With this aim, we use MDE (Model Driven Engineering) [8] to formally define the syntax. Models are described using a modelling language (e.g., UML), whose syntax is defined through a meta-model. Particularly, we represent the GDPR as a DSL (Domain Specific Language) [16], an excerpt depicted in Fig. 2. This DSL can be applied to any application domain.

D. Implementation and Testing

During these activities, the design is translated into code and the tests are performed to validate and verify that the implementation is valid according to the requirements. From the GDPR viewpoint, those activities are straightforward to carry out, since legal requirements have been already fulfilled at early stages. Fig. 3 illustrates all the activities involved in these phases. The implementation shall develop user interfaces displaying users’ rights and collecting their explicit consent, which shall be expressed voluntarily, specifically, and unambiguity by means of a clear affirmative action (Art. 6). For instance, it could be implemented using a pop-up window with this information. It is noteworthy that in no case the user consent can be marked as accepted by default (Art. 90). That is, if there is a checkbox for the acceptance, its default value must be unchecked. There are other means of obtaining the user consent, such as digital certificates, electronic signatures, or electronic national ID issued by...
certification authorities. As mentioned in analysis activities, the user interface shall validate user’s age. The processing of data of a child under fourteen shall be lawful only if it has the consent of their legal custodians, so it will be necessary to collect somehow such consent (Art. 7). In the case of not obtaining this consent, the application may have a restricted functionality. As a general rule, the application or website shall include a disclaimer reporting its owner, the privacy policy on which the following is communicated: the data processing, its purpose, if you have recipients and the identity and address of the person responsible for the treatment. Finally, it shall display the rights of the user and, if necessary, the cookie policy if they are used.

E. Maintenance

Once the product is delivered to the customer, the next activity is maintenance, which will be carried out by either the development company or a software maintenance company. In this activity. The activities carried out during this phase are summarised in Fig. 4. The European GDPR does not apply to the processing of personal data of deceased persons or of legal persons, therefore they must be removed. Nevertheless, the Spanish LOPD authorises to exercise the rights to access, of rectification and erasure with respect to the personal data of deceased persons (Art. 3). The role of the data controller will participate in case it is necessary to block those personal data (Art. 32). In the event this unlocking does not occur, the data controller will eventually have to destroy the personal data. Moreover, the data controller will be in charge of registering all those activities occurring on the data (Art. 31), in order to reliably monitor that they are correctly processed in the event that such information is requested. One of the key issues of data quality is accuracy, that is, personal data will be exact and, if necessary, updated (Art. 5). During the maintenance, the software product must carry out tasks to validate these premises, including the automatic erasure of personal data once the limit time has reached, periodic updating the information and revision of invalid data, uncompleted or void.

IV. Related Work

We reviewed the literature in search of recent related work on the scope of privacy requirements definition and management in software process activities. The goal of the literature search was to answer the following question: How are the articles of the European GDPR adopted in the software development process? From the analysis of the selected studies, we classified the related work into two categories depending on the type of software process activity addressing data privacy issues:

- Adoption of the GDPR in early development activities, like requirements elicitation, analysis and specification, and software design. This addresses the problem of validating compliance with the regulation a priori before starting and during software system development.
- Adoption of the GDPR in later development activities, such as system testing and maintenance. This addresses the problem of validating compliance with the regulation when the software system is in use.

With respect to papers dealing with GDPR issues in early development activities, Meis and Heisel [19] report a systematic extension of the problem-based privacy analysis method (ProPan) designed to identify software system privacy needs based on a set of functional requirements. Based on studies published from 2009 to 2019, Dias et al. [12] reported a systematic literature review (SLR) on software privacy and privacy requirements and the methodologies and techniques that are used for their elicitation and specification. The methodologies include LINDDUN, SQUARE for Privacy, and PriS, among others. The SLR results revealed that ICT practitioners are not altogether familiar with software privacy, privacy requirements and the Brazilian LGPD, which is an obstacle to the application of laws and directives governing data privacy. Amorim et al. [3] suggest the use of gamification techniques as an option for providing practitioners at an organization with data privacy training. Mavroedi et al. [18] also investigated the use of gamification for privacy requirements elicitation and engagement with the users. Perera et al. [22] proposed a guide based on the Privacy by-Design framework including a set of best practices to help software engineers to ensure user data privacy during the development of Internet of Things (IoT) applications. Rabinia et al. [24] highlight the difficulty to model the GDPR. This process tends to output models that are extremely complex and hard to understand due to the number of articles of which they are composed, as well as the complexity of both the articles that they contain and the issues that they address. They propose the use of the Formal Legal GRL framework, associated with a methodology to help address the complexity of the models and automate the modelling process. They focus exclusively on design, that is, they do not account for other development phases. BlancoLainé et al. [6] underscore the importance of the GDPR for businesses, which have difficulty understanding the legal requirements. They should take utmost care to ensure compliance, as any mistake can have an impact at all business levels. They attempt to facilitate this process by using enterprise architecture models to represent the GDPR regulation. In addition, there are GDPR-related papers [15] that primarily focus on the process of validating compliance with regulations once the system or application is in use, developing models to automate this process to enable any business, organization or even person to check that its systems or applications comply with the regulations and avoid possible penalties [4][26]. Torre et al. [26] propose analysing articles iteratively in search of keywords to help identify different artefacts and their relationships in order to model GDPR in a machine-readable format [25]. Other approaches, like Ayala-Rivera and Pasquale’s GuideMe [4], set out a systematic stepwise approach. They set out six stages in which to analyse the status...
of an organization or application, and plan and implement cor-
rective actions to fix non-compliant issues. This is a corrective
method for application on existing applications. It is not, there-
fore, suitable for use throughout the entire software development
process but is rather confined to the maintenance activity during
which most corrections are usually made. Besides the more
functional approaches outlined above, there are a series of best
practice guides on compliance developed by different companies
and organizations [1]. The EU [14] and companies like Deloitte
[11], Norton Rose Fulbright [20] have their own checklists and
benchmarks that they make available to their customers with a
view to establishing a series of general guidelines enabling a
company, organization or even a self-employed worker to assure
that their services, applications and infrastructures comply with
the regulations.

V. CONCLUSIONS AND FURTHER WORK

We propose a common reference framework to drive the in-
tegration of privacy guidelines into the software development
process and guarantee personal data privacy. In particular, we
provide a reference framework for adopting the articles of the
GDPR in the software process and integrate the legal require-
ments into all the Unified Process activities. Specifically, we
adapted each and every one of the 97 articles of the Spanish
Law on the Protection of Personal Data and Guarantee of Digital
Rights to the Unified Process. This reference framework consti-
tutes comprehensive guidance designed to familiarize a de-
velopment team with the legal requirements of the regulations
throughout all the software development activities, from require-
ments elicitation to deployment to customers and maintenance.
It also provides software development teams with a mechanism
for integrating the legal requirements into all the development
activity groups of the Unified Process. The development team
should include a legal expert in order to take into account all the
legal slants and details of specified articles that may be omitted
during the use of the reference framework in particular software
development projects. One example would be special types of
scientific research, historical or medical data processing, which
may be especially important in some software projects. A corpus
of legal terms shall be developed with experts in natural lan-
guage processing and legal corpus. Our future research sets out
to automate the reference framework advocating the integration
of the above professional profiles into software development
teams in order to apply natural language processing techniques
and automate the reference framework.

ACKNOWLEDGMENT

Work funded by the Spanish Ministry of Science (RTI2018-
095255-B-I00) and the R&D programme of Madrid
(P2018/TCS-4314).

REFERENCES

[1] Agencia Española de Protección de Datos (AEDP). Guía de Personal
Data Protection for Processing Controllers (In Spanish, Guía del Regla-
mento General de Protección de Datos para Responsables de Tra-
09/guia-rgrp-para-responsables-de-tratamiento.pdf

[2] Agencia Española de Protección de Datos (AEDP). Regulatory Com-
pliance List (In Spanish, Lista de cumplimiento normativo), November
tado-de-cumplimiento-del-rgrp.pdf

and security in cyberspace: Training perspectives on the personal data

[4] V. Ayala-Rivera and L. Pasquale. The grace period has ended: An ap-
proach to operationalize GDPR requirements. In Proc. of the IEEE 26th

odology to develop privacy-friendly software applications. Inf. Sci.,

architecture model for GDPR compliance principles. In Proc. of 12th IFIP
Working Conf. of The Practice of Enterprise Modeling, (PoEM’19), vol.

Protection and Digital Rights Guarantees (In Spanish, Ley Orgánica
3/2018, de Protección de Datos Personales y Garantías de los Derechos
2018-16673-consolidado.pdf

neering in Practice, Second Edition. Synthesis Lectures on Software En-

[9] A. Cavoukian. Understanding how to implement privacy by design, one


https://www2.deloitte.com/us/en/pages/risk/articles/deloitte-gdpr-bench-
marking-survey-the-time-is-now.html

P. H. Teixeira Costa, and F. Lima. Perceptions of ICT practitioners re-

tion Regulation), May 2016. https://eur-lex.europa.eu/legal-conten-
t/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN


[17] I. Jacobson, G. Booch, and J. Rumbaugh. The Unified Software Develop-

[18] A. Mavroedi, A. Kitisou, and C. Kalloniatis. The role of gamification in
privacy protection and user engagement. 2020.


fulbright.com/en/knowledge/publications/3614a527/gdpr-checklist


[23] PricewaterhouseCoopers US. GDPR Compliance Top Data Protection
press-release.html

yogy for implementing the formal Legal-GRL framework: A research pre-

[25] D. Soltes. Social networks as the best instrument for achieving full, world-
wide and truly global e-inclusion. Proc. 16th World Multi-Conference on

Goes. Using models to enable compliance checking against the GDPR:
An experience report. In Proc. of the ACM/IEEE 22nd Int. Conf. on
Model Driven Engineering Languages and Systems (MODELS’19).
Remaining Activity Sequence Prediction for Ongoing Process Instances

Xiaoxiao Sun*, Yuke Ying, Siqing Yang and Hujun Shen
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China

sunxiaoxiao@hdu.edu.cn, yingyuke_hdu@163.com, yangsiqing@hdu.edu.cn, 191050007@hdu.edu.cn

Abstract—Remaining activity sequence prediction (i.e., Activity suffix prediction) aims at recommending the most likely future behaviors for ongoing process instances (traces), which enables process managers to rationally allocate resources and detect process deviations in advance. Recently, techniques of neural networks have found promising applications in activity suffix prediction by training a prediction model for next activity and iteratively performing the model to achieve the whole sequence prediction. However, the iterative prediction accumulates the deviations of each iteration and the result also lacks interpretability. In this paper, we propose a novel method to predict activity suffixes from the perspective of control flow and data flow for ongoing traces, where process discovery and trace replay techniques are employed to simulate executions of traces under real conditions and Long Short-Term Memory (LSTM) is applied to characterize the correlation between executed information and future execution. Sequence matching between historical prefix traces and ongoing traces are performed based on the above information to select the optimal-matched (i.e., most similar) activity suffix for ongoing process instances. Experiments on real-life datasets demonstrates that our proposed method outperforms other methods.

Keywords—activity suffix prediction; process discovery; trace replay; LSTM; sequence matching

I. INTRODUCTION

Business process management (BPM) is a technique that modifies and extends business processes for enterprises by continuously mining, modeling, and monitoring process instances (traces) [1]. As a concrete practice that supports the businesses of enterprises, Process Aware Information System (PAIS) [2] record execution information of process instances (i.e., event logs) such as activity (i.e., event type), timestamp, resources and so on, which is further analyzed for process optimization and improvement. Traditional process mining pays attention to offline analysis like process discovery, conformance checking between logs and process models while recent focus of researchers has gradually turned to online analysis, especially predictive business process monitoring (PBPM). PBPM dedicates to predicting future execution information for ongoing traces such as next activity, remaining time, final outcome, remaining activity sequence (i.e., activity suffix) and so on, which provides reference information for process executors and helps process managers to take effective measures for optimizing process executions.

Among all prediction tasks of PBPM, next activity prediction and remaining time prediction are the most widely studied tasks while activity suffix prediction is seldom considered as a single theme. However, compared to next activity prediction and remaining time prediction, activity suffix prediction provides more extensive future information for both process executors and process managers. Accurate prediction of activity suffix helps process managers to perceive early deviations and resource shortages, which can be prevented by timely and effective measures.

Recently, techniques of neural networks especially Recurrent Neural Network (RNN) [3] and Long-Short Term Memory (LSTM) [4] are widely employed in the fields of PBPM. These works address the tasks of PBPM as regression problems or classification problems and utilize historical traces to train prediction models. However, the diversity of activity suffix categories makes this solution inapplicable. Currently, a variety of researches achieve activity suffix prediction by constructing a prediction model for next activity and iteratively performing the model to predict the whole sequence. These methods, however, are super sensitive to hyper-parameters and each iteration would add deviations to the final result.

Therefore, in this paper, we propose a novel sequence-matching-based approach from the perspective of both control flow and data flow for activity suffix prediction. Summarily, the main contribution of this paper is as follows:

- The techniques of process discovery and trace replay are applied to simulate the real behavioral context of trace executions.
- LSTM is employed to train a prediction model that implicitly characterizes the correlations between executed information and future execution.
- Sequence matching is performed between historical prefix traces and ongoing traces based on the above information to obtain the most similar activity suffixes for ongoing traces.

The rest of this paper is structured as follows: Section II introduces existing works on activity suffix prediction. A detailed description of our approach is presented in Section III. Section IV evaluates the effectiveness of our method and conducts comparisons with the optimal result of other researches on real-life datasets. Eventually, conclusions and future work are demonstrated in Section V.

DOI reference number: 10.18293/SEKE2021-075
II. RELATED WORK

A variety of researches have been put forward to realize activity suffix prediction in the past decade, which are roughly divided into two types according to whether process structures are extracted from event logs, i.e., process-aware methods and non-process-aware methods [5].

Process-aware methods for activity suffix prediction require constructions of process models such as petri net and then apply the models to accomplish prediction. For example, Spoel et al. [6] first mine a causality graph from the event log. Then, they adjust and apply a well-known shortest path algorithm (i.e., Floyd-Warshall algorithm) over the mined causality graphs to find a path whose sum of weights is the least. Similarly, Polato et al. [7] construct a transition system based on the event log and annotate its edges with transition probabilities. Then, they define a cost (i.e., the opposite of the logarithm of the transition probability) between two nodes and apply a shortest path algorithm on annotated transition systems for activity suffix forecasting.

Recently, neural networks especially RNN and LSTM are widely applied to achieve activity suffix prediction, which are typical non-process-aware methods. Tax et al. [8] and Evermann et al. [9] employ LSTM to forecast activity suffixes for ongoing traces while Lin et al. [11] apply RNN to achieve activity suffix prediction. The similarity of their works is that they all iteratively forecast next activity to realize activity suffix prediction. Specifically, Tax et al. [8] construct a prediction model to forecast both the type of the next event (i.e., next activity) and its timestamp at the same time using a shared LSTM layer. Evermann et al. [9] realize the similar works as [8], however, they encode attributes via embedding space instead of one-hot encoding. Lin et al. [10] propose a RNN-based predictive model called MM-Pred to predict next activities and related attributes, and conduct iteration of the model to obtain the remaining event sequence. In addition, Taymouri et al. [11] present an encoder-decoder architecture grounded on Generative Adversarial Networks (GANs), which generates a sequence of activities and their timestamps in an end-to-end way.

Summarily, most of the current researches adopt neural networks to accomplish activity suffix prediction, which lacks deep mining on event logs as well as interpretation of the predictive results. Therefore, in this paper, we attempt to propose a novel approach to accomplish the task, where process discovery and trace replay techniques are employed to simulate the real trace execution environment and LSTM is applied to characterize the correlation between executed information and future execution of ongoing traces. Eventually, sequence matching is applied to achieve the final prediction based on the above information.

III. APPROACH

A. Preliminaries

1) Event logs

Definition 3.1 (Event; Trace; Event log). An event is one single execution of an activity in different contexts, represented as \( e = (c, a, t, t_e, r, d_1, ..., d_m) \), where \( c \in C \) is the case to which the event belongs, \( a \in A \) represents the associated activity, \( r \in R \) is resources required for execution, \( t \) and \( t_e \) represent the start and end timestamp respectively, and \( d_1, ..., d_m \) represent the other basic attributes. A trace is a finite ordered sequence of events expressed as \( \sigma = \langle e_1', e_2', ..., e_n' \rangle \), where \( |\sigma| \) is the length of \( \sigma \). An event log \( L \) is a collection of multiple traces, which is expressed as \( L = \{\sigma_1, \sigma_2, ..., \sigma_n\} \), and \( |L| \) denotes the number of traces in \( L \).

Definition 3.2 (Prefix Trace, PT; Suffix Trace, ST). A PT is the first \( k \) events of a trace \( \sigma \), which is denoted as \( PT_k(\sigma) = \langle e_1', e_2', ..., e_k' \rangle \). Correspondingly, a ST is the last \( r \) events of trace \( \sigma \) and is represented by \( ST_r(\sigma) = \langle e_{r+1}', e_{r+2}', ..., e_n' \rangle \).

Definition 3.3 (Activity Sequence, AS). Given a trace \( \sigma \), its activity sequence is composed by activities of its events, which is expressed as \( AS(\sigma) = \langle e_1', a, e_2', a, ..., e_n', a \rangle \).

Definition 3.4 (Event encoding; Encoded matrix). An event encoding is a function \( f : e \rightarrow D^m \) that transforms the attribute values of event \( e \) into a numerical vector (one-hot encoding for category attributes and normalization for numeric attributes), where \( m \) denotes the dimension of the encoded vector. Then, for each trace \( \sigma \), we integrate encoded vectors of its events by time order and obtain an encoded matrix expressed as \( EM(\sigma) = [f(e_1'), f(e_2'), ..., f(e_n')] \).

2) Petri net

Definition 3.5 (Petri Net). A petri net is an explicit representation of an event log consisting of nodes (places and transitions) and direct arcs. Each place holds a non-negative integer number of tokens, which can be transferred according to firing rules (Definition 3.7). The number of tokens in place \( p_i \) is expressed as \( \beta(p_i) \). A petri net is defined as a six-tuple, i.e., \( pn = (P, T, F, A, \pi, M) \), where:

- \( P = \{p_0, p_1, ..., p_{|P|-1}\} \) is a finite and non-empty set of places.
- \( T = \{t_0, t_1, ..., t_{|T|-1}\} \) is a finite and non-empty set of transitions. Transitions in petri net are associated with activities of an event log by a function \( \pi \), that is, \( a = \pi(t) \), \( t \in T \), \( a \in A \cup \{\tau\} \), where \( A \) is the activity set of an event log and \( \{\tau\} \) represent non-observable activities. Transitions interrelated to non-observable activities are hidden (invisible) transitions.
- \( F \subseteq (P \times T) \cup (T \times P) \) is the set of directed arcs connecting places and transitions.
- \( M \) is the marking that represents the state (the token distribution of places) of the petri net and is denoted as \( M = [\beta(p_0), \beta(p_1), ..., \beta(p_{|P|-1})] \), where \( \beta(p_i) \) can be expressed as \( M[i] \), \( 0 \leq i < |P| \).
Definition 3.6 (Input Set, Output Set). Given a node \( x \in P \cup T \), its input set is denoted as \( \bullet x = \{ y | y \in P \cup T \land (y, x) \in F \} \) and its output set is represented by \( \bullet x = \{ y | y \in P \cup T \land (x, y) \in F \} \).

Definition 3.7 (Firing rules). A transition \( t \in T \) is enabled iff \( \forall p_i \in \bullet t : \beta(p_i) > 0 \). Besides, when the transition \( t \) is enabled, it can be fired and current marking \( M \) converts into a new marking \( M' \), where \( M'[i] \) is calculated as:

\[
M'[i] = \begin{cases} 
M[i] - 1, & p_i \in t \\
M[i] + 1, & p_i \in \bullet t \\
M[i], & \text{otherwise}
\end{cases}
\]

(1)

B. Sequence-matching-based activity suffix prediction

After presenting basic concepts and definitions of this paper, this section introduces the procedure of sequence-matching-based activity suffix prediction, which is divided into three parts, i.e., behavioral context replay, data context prediction and sequence matching.

1) behavioral context replay

Trace replay is a technique that executes traces of an event log on a process model to measure the conformance between the event log and the model [1]. In this paper, inspired by Theis et al. [12], we develop a new application of trace replay to simulate the real-life environment of process execution, i.e., behavioral context replay. Since rare behaviors cannot be characterized by the process model, we adjust the firing rules in Definition 3.7 to guarantee that all the transitions related to activities of traces to be enabled. Specifically, when a transition \( t \) is not enabled, we first obtain its input set and find places with token missing. Then, to fill requirements of tokens in these places, we further enable some hidden transitions that connects these places with other places which hold tokens. If \( t \) still cannot reach the enabled state by the above operations, we manually add tokens to these places to fulfill the firing requirement of transition \( t \).

In detail, we first conduct process discovery on an historical event log to obtain a petri net using Inductive Miner (IM), which is easy to operate and friendly to implement trace replay [13]. Then, for each trace \( \sigma \) in the event log, we replay it on the obtained petri net according to the adjusted firing rules to simulate its execution. Specifically, during the replay of \( \sigma \), whenever a transition \( t \) related to activity \( a_i \) is fired, we update the token value of each place and acquire a new marking \( M_t \). When the trace ends its replay, we integrate all markings and obtain the behavioral context information of \( \sigma \), which is denoted as \( \text{BehavContext}(\sigma) = \{M_{t_1}, M_{t_2}, \ldots, M_{t_{|\sigma|}}\} \). The whole process of behavioral context replay is illustrated in Figure 1.

To measure the behavioral context consistency between two traces, we further introduce a definition named trace behavioral similarity (TBS), whose mathematical expression is illustrated as (2).

Definition 3.8 (Trace Behavioral Similarity, TBS). Given two traces \( \sigma_1 = \langle e_{10}^1, e_{10}^2, \ldots, e_{|\sigma_1|}^1 \rangle > \) and \( \sigma_2 = \langle e_{10}^2, e_{10}^2, \ldots, e_{|\sigma_2|}^2 \rangle > \), their TBS is defined as:

\[
TBS(\sigma_1, \sigma_2) = \frac{\sum_{i=1}^{|\sigma_1|} \beta(p_i)}{|\sigma_1| \times \max(|\sigma_1|,|\sigma_2|)}
\]

(2)

Where \( \beta(p_i) \) represents the token equivalence of place \( p_j \) after the i-th activity is executed. The average equivalence of \( \text{BehavContext}(\sigma_1) \) and \( \text{BehavContext}(\sigma_2) \) is calculated as \( TBS(\sigma_1, \sigma_2) \). Besides, if the dimension of \( \text{BehavContext}(\sigma_1) \) and \( \text{BehavContext}(\sigma_2) \) is not consistent, we stuff the one with smaller dimension using padding vectors, i.e., vectors filled with 0.

2) Data context prediction

During the execution of process instances, a variety of data information is produced and recorded as attributes (i.e., resource, cost and so on) in event logs, which is collectively considered as data context in this paper. Data context characterizes the variations of essential attributes, which has significant influence on future execution. In this section, we attempt to mine the correlation between executed information and future execution, and use LSTM model to predict the future data context of ongoing traces.

In detail, we first split historical traces in event logs into PTs and STs. Then, several time-related features are added including year, month, day, weekday, hour and duration for the purpose of enriching information. Subsequently, we perform event encoding on PTs and STs, whose encoded matrixes are considered as the input and training target of LSTM respectively to learn their correlation. Meanwhile, to reduce memory consumption during the training, we conduct dimensionality reduction on suffix matrixes using a popular technique named Uniform Manifold Approximation Projection (UMAP) [14] before training, which is used to deal with high-dimensional data. The matrix of ST st after dimensionality reduction is denoted as \( E_{\text{UMAP}}(st) \). After training, the correlation between executed data information and future execution is implicitly expressed in the prediction model. For an ongoing trace, we import its encoded matrix to the prediction module.
model and the output is the predicted data context we need for further sequence matching.

3) Sequence matching

After introducing the behavioral context replay and data context prediction, we further describe the procedure of sequence matching in this section, which is divided into four steps:

Step 1: To better simulate real-life executions, we sort traces in event logs by time and take the first 70% of traces as training set and the remaining 30% of traces as testing set. We further divide the traces in training set and testing set into PTs and STs, where PTs and STs of training set are employed for sequence matching while PTs of testing set are considered as ongoing traces and STs are utilized for evaluation.

Step 2: Subsequently, we perform process discovery on training set using IM and obtain a petri net. For each PT pt in training set and testing set, we replay it on the petri net and acquire its behavioral context information BehavContext(pt).

Step 3: Traces in training set is applied to train the prediction model for data context as mentioned above. Then, for each PT pt in testing set, we import its encoded matrix into the prediction model and obtain its predictive data context, which is denoted as DataContext(pt).

Step 4: After step 2–3, each PT pt in testing set is associated to BehavContext(pt) and DataContext(pt). Then, we perform sequence matching between PTs in testing set and PTs in training set. Specifically, for each PT pt in testing set, we traverse PTs in training set and select PTs with the highest TBS as pt. Then, we further calculate the Euclidean Distance (ED) between the data suffix matrix es of selected PTs and TBS as.

Algorithm 1: The procedure of Step 4.

**INPUT:** 1. Training set trainingS 2. A PT pt in the testing set

**OUTPUT:** The predictive activity suffix of pt, PAS(pt)

**BEGIN**

01: maxTBS ← 0; minDist ← ∞; PAS(pt) ← null;
02: FOREACH trace σ in trainingS DO:
03: FOREACH i in range(1, |σ| +1) DO:
04: curTBS ← TBS(pt, PT(σ))
05: IF curTBS > maxTBS DO:
06: maxTBS ← curTBS; PAS(pt) ← AS(TSpt,σ));
07: minDist ← ED(UMAP(TSpt,σ), DataContext(pt));
08: ELIF curTBS == maxTBS DO:
09: curDist ← ED(UMAP(TSpt,σ), DataContext(pt));
10: IF curDist < minDist DO:
11: minDist ← curDist; PAS(pt) ← AS(TSpt,σ));
12: RETURN PAS(pt);
**END**

**Sepsis:** This real-life event log contains events of sepsis cases from a hospital, which were recorded by the Enterprise Resource Planning (ERP) system.

**BPIC2012W_Complete:** BPIC2012 dataset is an event log taken from a Dutch Financial Institute and represents the process of an application process for a personal loan or overdraft within a global financing organization, which can be split into three sub-processes, i.e., the application itself (BPIC2012A), the work items belonging to applications (BPIC2012W) and the offer (BPIC2012O). In this paper, events with the transition lifestyle of “completed” in BPIC2012W are employed to conduct experiments, which is called BPIC2012W_Complete.

**BPIC2012W_Deduplication:** Since BPIC2012W_Complete contains a lot of self-loops, i.e., some activities are continuously executed several times, we further perform experiments on BPIC2012W_Complete without self-loops, which preserves the first loop of traces and removes the others. The processed dataset is named as BPIC2012W_Deduplication.

**TABLE I. CHARACTERISTICS OF DATASETS**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Trace</th>
<th>#Event</th>
<th>#Activity</th>
<th>#AS</th>
<th>#Avg. length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Helpdesk</td>
<td>4580</td>
<td>21348</td>
<td>14</td>
<td>226</td>
<td>4.66</td>
</tr>
<tr>
<td>Sepsis</td>
<td>1050</td>
<td>15214</td>
<td>16</td>
<td>895</td>
<td>13.64</td>
</tr>
<tr>
<td>BPIC2012W_Complete</td>
<td>9658</td>
<td>72413</td>
<td>6</td>
<td>2263</td>
<td>7.50</td>
</tr>
<tr>
<td>BPIC2012W_Deduplication</td>
<td>9658</td>
<td>29410</td>
<td>6</td>
<td>71</td>
<td>3.05</td>
</tr>
</tbody>
</table>

**B. Evaluation metrics**

Similar as references [8], [10] and [11], we employ Demerau-Levinstain similarity (DLS, DLS ∈ [0,1]) to measure the similarity of the true activity suffix (TAS) and the predictive activity suffix (PAS). The mathematical representation of DLS is illustrated as (4), where DL(PAS,TAS) is the Demerau-

*B. Evaluation metrics*
Levenshtein distance between $PAS$ and $TAS$, and $PAS.length$ and $TAS.length$ represent the length of $PAS$ and $TAS$ respectively. Demerou-Levenshtein is the minimum number of single-character editions (i.e., insertion, deletion, substitution, and transposition) required to transform one sequence into another.

$$DLS(PAS,TAS) = 1 - \frac{DL(PAS,TAS)}{\text{max}(PAS.length,TAS.length)}$$  \hspace{1cm} (4)

C. Experimental setup

Our experiments were run on a 10 core Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz with 64 GB RAM. The approach was implemented in Python 3.6, Keras 2.2.4 with Tensorflow 1.15.0 backend and Pm4py 2.1.0, using CUDA 10.1 and UMAP 0.5.1. The hyper-parameters of LSTM and UMAP are shown in Table II, where $\text{maxLen}$ is the length of the longest trace in the log.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Value</th>
<th>Hyper-parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>LSTM layers</td>
<td>1</td>
<td>Epoch</td>
<td>200</td>
</tr>
<tr>
<td>LSTM units</td>
<td>50</td>
<td>Dropout</td>
<td>0.5</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
<td>$n_{neighbors}$ (UMAP)</td>
<td>5</td>
</tr>
<tr>
<td>Batch size</td>
<td>128</td>
<td>$\text{min_dist}$ (UMAP)</td>
<td>0.3</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.001</td>
<td>$n_{components}$ (UMAP)</td>
<td>$\text{maxLen}$</td>
</tr>
<tr>
<td>Loss</td>
<td>mse</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

D. Result

Table III summarizes the performance of our method on four datasets in terms of the average DLS. We further analyze the performances in three specific prefix lengths as short PTs, medium PTs and long PTs. As shown in the table, we calculate the average DLS of PTs whose length is more than 2, 4 and 6 for Helpdesk and BPIC2012W_Deduplication while calculate average DLS of PTs whose length is more than 2, 5 and 10 for the other two datasets since the sequence length of Helpdesk and BPIC2012W_Deduplication is relatively shorter than the other two. Besides, All represents the average DLS of all PTs in the event log. From the table, we notice that Helpdesk achieve the best DLS, i.e., 84.01%, while BPIC2012W_Complete demonstrates relatively poor performance.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>helpdesk</td>
<td>0.8585</td>
</tr>
<tr>
<td>Helpdesk</td>
<td>0.3402</td>
</tr>
<tr>
<td>BPIC2012_W_Complete</td>
<td>0.2936</td>
</tr>
<tr>
<td>BPIC2012W_Deduplication</td>
<td>0.4575</td>
</tr>
</tbody>
</table>

Figure 2. The variation trends of DLS and CD in different datasets.

Furthermore, to explore the reason for the difference of performances among datasets, we introduce a definition named Coincidence Degree (CD).

**Definition 4.1 (Coincidence Degree, CD).** The CD of traces in an event log is defined as (5), where $\#AS$ and $\#Trace$ mean the number of different activity sequences and the total count of traces in the log, respectively. The trace behavior of event logs with low CD are highly variable, which improves the difficulty of sequence matching.

$$CD = 1 - \frac{\#AS}{\#Trace}$$  \hspace{1cm} (5)

We analyze the correlation between CD and DLS for four datasets, which is shown in Figure 2. In general, our method demonstrates better performance in datasets with high CD while performs relatively poor in datasets with low CD. For example, the CD of Helpdesk is high and its average DLS is correspondingly high while Sepsis demonstrates an opposite situation. However, we notice that the CD of BPIC2012W_Complete is high while its average DLS is low. As mentioned above, this dataset contains a lot of self-loops, which causes our approach to predict overly long sequences of the same activity. From the figure, we conclude that the average DLS of BPIC2012W_Deduplication improves a lot compared to BPIC2012W_Complete, which demonstrates that the self-loops have an adverse effect on our prediction.
Implementation | Helpdesk | BPIC2012W Complete | BPIC2012W Deduplication
--- | --- | --- | ---
Our method | 0.8585 | 0.8946 | 0.4575
Tax et al.[8] | 0.7669 | - | 0.3937
Evermann et al.[9] | - | - | 0.2970
Lin et al.[10] | - | - | -
Taymouri et al.[11] | 0.8411 | - | -

Note: "-" represents that the corresponding result of the dataset is not reported in the reference.

**Figure 3.** The average DLS at different prefix lengths of four datasets.

### 5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a method to address the problem of activity suffix prediction, where process discovery and trace replay techniques are employed to simulate executions of traces under real conditions and LSTM is applied to predict and characterize future data context of ongoing process instances. Besides, the above information is eventually applied to perform sequence matching between historical PTs and the current traces. The result of our method outperforms the best result of most methods.

Since our work only cope with the problem of activity suffix prediction, we plan to make suffix predictions related to other execution status such as resource, time and so on in the future, which would provide more reference information for both process executors and process managers.

**ACKNOWLEDGMENT**

This work is supported by Natural Science Foundation of China (No.61472112), Natural Science Foundation of Zhejiang Province (No.LQ20F020017) and the Key Science and Technology Project of Zhejiang (No.2017C01010).

**REFERENCES**


Studying the Impact of the User Subscription Times in Different Cloud Configurations

Hernán-Indibil De la Cruz
HernanIndibil.Cruz@uclm.es

María-Emilia Cambronero
MMarilia.Cambronero@uclm.es

Valentín Valero
Valentin.Valero@uclm.es

Pablo C. Cañizares
pablo.cerro@uam.es

Adrián Bernal
Adrian.Bernal@uclm.es

Alberto Núñez
Alberto.Nunez@pdi.ucm.es

1 Albacete Research Institute of Informatics, Computer Science Department, Universidad de Castilla-La Mancha
2 Computer Science Department, Autonomous University of Madrid
3 Software Systems and Computation Department, Complutense University of Madrid

Abstract

In this paper, we model cloud systems and the user interactions with the cloud provider using the UML2Cloud profile. In general, users request virtual machines according to their needs, but they can also subscribe to the cloud provider and wait to be notified when the requested resources are not available. In this case, users indicate a maximum subscription time, so once this time elapses without being notified, users leave the system unattended. In this paper, then, we present an exhaustive research study to measure how the user subscription times affect the overall system responsiveness. In this study, three different cloud configurations are analyzed. Each cloud processes several workloads, which are generated using two distribution functions for the user arrivals, namely a normal and a cyclic normal distribution. The purpose of this study is to find out the inflection point for the waiting time of the users, from which the cloud responsiveness and its performance do not improve. The obtained information is therefore useful for the cloud provider to improve the configuration of the cloud.

1. Introduction

Cloud computing is experiencing important growth nowadays. Cloud service providers need tools that allow them to better manage their resources, with the goal of maintaining the Quality of Service offered to a growing number of customers, agreed in the so-called Service Level Agreements (SLAs). One of these tools is the simulation and, particularly, cloud simulators, which allow us to simulate workloads that are executed in virtual environments. With these tools, we can predict behaviors in the real cloud systems, even before these systems are built and deployed so that they allow cloud providers to anticipate some problems that could arise once the system is running.

In addition, modeling the cloud infrastructure and the users’ interactions with the cloud providers allow us to have a better understanding of the behavior of all the roles in these systems. With this purpose in mind, we defined the UML2Cloud profile [1]. The main features of the cloud infrastructure, that is, CPUs, storage, and network bandwidth, among others, are considered in this parameterized profile, as well as the exchange of messages between the users and the cloud provider, with parameters such as the specification of the virtual machines required, the applications to be executed on them, and the maximum subscription time when the requested machines are not available at the time of the request.

This paper aims at studying the behavior of simulated cloud environments modeled with the UML2Cloud UML profile. In essence, this study focuses on the abandon-rate and waiting time of the users, in order to help finding the best configurations and workloads for the analyzed cloud systems. We investigate the relationships between the number of users trying to be served by the cloud and the waiting time due to users’ subscriptions, which is a quality of service-related metric defined in the UML2Cloud profile.

There are several works in the cloud literature studying different resource allocation policies with the goal to meet the quality of service (QoS) features. For instance, Kouki et al. [6] present an analytical performance model to predict cloud service performance taking into account the last values for abandon rate, latency, and cost. Following the same line, Wu et al. [12] propose several resource allocation algorithms for SaaS providers to minimize SLA violations and infrastructure costs by managing the workloads. Mateo-Fornés et al. [7] present an analytic model, called CART, for studying cloud availability and response time to improve several QoS items, as performance, cost, and availability in SaaS. There are significant differences with our work, in which we consider the publish-subscribe paradigm and we study the impact of user subscription times on QoS parameters, like response time and performance.

DOI reference number: 10.18293/SEKE2021-095
Vinodhini [11] analyzes a cloud system based on a queueing model with possible failures and cloud repairs, instead of the publish-subscribe paradigm used in this study.

There are other works that analyze different metrics related to performance evaluation. For instance, Yang et al. [13] evaluate cloud performance taking into account the service response time in an environment with fault recovery to improve cloud reliability and considering a queueing system to conduct the performance analysis. Similarly, Khazaei et al. [5] propose an approach also focused on the response time, using other queueing system model.

In general, these works are based on theoretical models and the obtained results are based on the assumptions that need to be established for the analysis of these models. An alternative is the usage of cloud simulation, which is a widely adopted technique that allows us to reproduce the behavior of real cloud environments. Furthermore, simulation allows mitigating some problems related to these environments, such as the experiments reproducibility and the high costs of renting real cloud systems.

In the current literature, we can find multiple proposals based on simulation tools to study different aspects of the cloud [3]. Some simulators focus on resource provisioning algorithms, such as CloudSim [2], and Network-CloudSim [4]. Another simulator, SimIC, is focused on the management of large-scale resources in inter-cloud environments. Finally, iCanCloud [9] helps users of a cloud deciding the best starting conditions on pay-as-you-go scenarios.

In this paper, we focus on the use of cloud simulation. Specifically, we use the Simcan2Cloud simulator [1], which is a simulation tool of parallel and distributed architectures and applications. We use a different perspective, our approach focuses on the users waiting time analysis when they subscribe to the cloud provider. Thus, users are notified when the resources they need are available. This metric is measured in a cloud environment close to saturation, i.e., in a cloud system where a high number of users are requesting resources, in comparison with the cloud size.

In this paper, we focus on the use of cloud simulation. Specifically, we use the Simcan2Cloud simulator [1], which is a simulation tool of parallel and distributed architectures and applications. We use a different perspective, our approach focuses on the users waiting time analysis when they subscribe to the cloud provider. Thus, users are notified when the resources they need are available. This metric is measured in a cloud environment close to saturation, i.e., in a cloud system where a high number of users are requesting resources, in comparison with the cloud size.

In this study, each cloud processes different workloads, which have been generated using two distribution functions for the users’ arrival, a normal and a cyclic normal distribution. Thus, we analyze the impact of the maximum subscription times in the cloud behavior, in terms of the requests that are finally served, the average waiting times for users, and the number of unattended users. Subscription time has been chosen as a key parameter in this study because it influences the trade-off between the waiting time and the number of unattended users. If users have a long subscription time, then the queue for resources is enlarged and the average waiting time increases as well. In contrast, users with a short subscription time will leave earlier unattended, keeping the queues with a lower number of users and the average waiting times will decrease.

The paper is structured as follows. Section 2 presents the background. Section 3 shows the methodology used to conduct the experimental phase of our study. In Section 4, a complete study about the impact of the user waiting time in different cloud configurations. And finally, Section 5 presents the conclusions and future work.

2. Background

In this section, we present an overview of the UML2Cloud profile [1], which has been created using UML, for the modeling of both cloud systems and the behavior of the users when they interact with the cloud provider. We also describe the Simcan2Cloud cloud simulator [1] that we use in the experiments.

2.1. The UML2Cloud UML profile

In this profile, a cloud system consists of a cloud provider, one or more data centers, and clients (also called cloud users) requesting resources to the cloud. The cloud provider manages a catalog of Virtual Machines (VMs) and hardware resources provided by the data centers. Each data center consists of a collection of physical machines, also called nodes, which are grouped by racks. Thus, each rack contains a set of nodes with the same hardware features, that is, CPU, memory, and storage. The whole cloud infrastructure is described in a component diagram, which can be found in our previous work [1].

The interactions between the users and the cloud provider are modeled using a sequence diagram (SD).

Figure 1 shows a new version of the SD presented in [1]. The interaction starts with a request message from the user, containing a list of all the VMs needed to execute its apps. Each VM is defined as a tuple: VM=(number, VM_type, renting_time) where we indicate the number of VMs of a certain type (VM_type) that we request, and the renting time.

The user then enters into a loop to handle the messages received from the cloud provider, until no requested VM is in execution or no subscription is active for this user. The answer to a request is a response message that contains the set of IPs corresponding to the physical machines containing all the requested VMs, which can be empty if this request cannot be attended to. If the set of IPs is not empty, the request can be served, and the user sends an execute message containing the list of applications (APPs) to execute and the list of IPs in which each APP is executed. Otherwise, when the set of IPs is empty, that is, at least one of the VM requested cannot be provided, the user can subscribe to the cloud provider indicating the VMs required and a maximum subscription time (subscribe message). The latter is the maximum time that the user is willing to wait for being
2.2. Simcan2Cloud Simulator

Simcan2Cloud [1] is a cloud simulator written in C++ using OMNeT++ [10], which is a cloud extension of SIMCAN [8], a tool for the simulation of parallel and distributed architectures and applications. The implementation of Simcan2Cloud fulfills the cloud specifications and the user interactions defined in the UML2Cloud profile. Simcan2Cloud is designed to provide a high level of flexibility, allowing the user to set up the cloud configuration in a modular way, in terms of data centers, computing and storage nodes, and network connections, among other components. Thus, Simcan2Cloud allows us to model and analyze different cloud scenarios.

3. Methods

In this section, we describe the methodology used to study the impact of the maximum user subscription time in different cloud configurations. In this study, we consider three cloud configurations, consisting of 64, 96, and 128 physical machines, respectively, where all machines have the same configuration (CPU and storage). These configurations have been chosen to analyze the impact of subscription times as we increase the number of nodes. Each cloud system processes several workloads, which are generated by establishing the inter-arrival time for the users, who execute the same application on the VMs. Each workload consists of 5000 users that request services to the cloud provider for a period of 5 days, where each VM is rented for 2 hours.

The first group of experiments analyzes the cloud responsiveness when users come at a normal distribution basis, with a single peak in the workload. In this case, a normal distribution with a mean of 3 days and a standard deviation of 1 day has been considered (see Figure 2a), where the x-axis represents the user arrival time and the y-axis shows the number of users. The second group of experiments analyzes the impact of daily burst user arrivals and, thus, a cyclic normal distribution has been considered (see Figure 2b). This figure represents the repetition of the same normal distribution in cycles of 24 hours of duration, with strong daily peaks at midday, where the x-axis represents the user ID and the y-axis shows the arrival time (in hours). In this case, therefore, we consider a normal distribution with a mean of 12 hours and a standard deviation of 3 hours.

The main goal of this study is to analyze the impact of the maximum subscription time that users establish when they

Figure 1: Cloud provider and user interaction SD.

Figure 2: Distributions of number of user arrivals per time intervals.
subscribe to the cloud provider. For simplicity, all the users assign the same value for the maximum subscription time, so the experiments are repeated using different values for this parameter. Thus, we put the focus on the waiting time obtained for the users when they intend to execute their applications in the cloud. The results obtained when different values for the maximum subscription time are used, provide us with valuable information about the responsiveness of the cloud and allows us to conclude the best configurations according to the submitted workload.

4. Results and Discussion

In this section, we show the results obtained from the empirical study described in Section 3. First, in Section 4.1 we show an experiment in which the cloud processes a user workload generated using a normal distribution. Next, in Section 4.2, we conduct an experiment in which the cloud processes a user workload generated using a cyclic normal distribution. Finally, we present a discussion of the obtained results in Section 4.3.

4.1. Case Study 1: Normal Distribution

In this scenario, users arrive by following a normal distribution with a mean of 3 days and a standard deviation of 1 day.

Figure 3 shows the results obtained for a cloud consisting of 64 physical machines, considering the following values for the maximum subscription time: 30, 50, and 70 hours. In these charts, the x-axis shows the user IDs, while the y-axis shows the waiting time for the users to be attended to. Black dots represent users that were fully served, while the red ones represent users that left the system without being served. The latter situation occurs when the maximum subscription time elapses and the cloud is not able to provide the user the requested resources. The first chart (left) shows the results obtained for a maximum subscription time of 30 hours. In this case – approximately – the first 1000 users are immediately served, i.e. their waiting time is 0. However, as more users arrive at the system, the cloud becomes more saturated and, approximately, when 2500 users are processed, the cloud cannot serve the new users’ requests, so they leave the system without being served (red dots at the upper area). Finally, once the user arrivals slow down after the peak, we can see that the final users are again served, but with a high waiting time.

When the subscription time is set to 50 hours (central figure) similar results are obtained. However, the point at which users leave the cloud is obtained when – approximately – 3700 users are attended. When the subscription time is set to 70 hours (right figure), we can see that the cloud can attend to all the requests, and the maximum waiting time is about 63 hours. This is the inflection point for the user waiting time, that is, the point at which the cloud responsiveness reaches the worst value and, at the same time, it can attend to all the users’ requests, so it should be the maximum subscription time for the users if they wish their works to be executed.

Figure 3: Case study 1 with 64 computing nodes.

Figure 4 shows the results for a cloud consisting of 96 physical machines, where the maximum subscription time ranges from 10 to 30 hours. This figure shows similar results to those obtained in the previous case. However, it is important to note that we have considered smaller values for the maximum subscription time. Thus, in this case, the inflection point is of 21.32 hours, so this should be the maximum subscription time for users that want their works to be executed.

Figure 4: Case study 1 with 96 computation nodes.

Figure 5 shows the results obtained for a cloud with 128 physical machines. In this case, we consider 2, 4, and 6 hours for the maximum subscription time. The first chart (left) shows how the cloud saturation appears at about user 2800, from which some users leave the cloud without being served. We notice that some users can actually be served in the upper part of the normal distribution as a consequence of the specific random numbers that were generated (see Figure 2a). However, in general, we would obtain a red line in the upper part of the figure. Finally, the final users can be attended to with better waiting times as in the previous cases. In the central chart, we can see the results for a maximum subscription time of 4 hours. In this case, most of the users can be served, and only a few of them must leave the cloud being unattended. When 6 hours are considered as maximum subscription time, all the users are served. In fact, the inflection point for the user waiting time is of 5.16 hours (see Table 1).

4.2. Case Study 2: Cyclic Normal Distribution

In this experiment, we consider a workload in which the users arrive following a cyclic normal distribution. Thus,
the workload has 5 peaks, so every day at midday we have – approximately – 1000 users requesting services to the cloud, with a peak of approximately 700 users.

Figure 6 shows the results of a cloud consisting of 64 physical machines processing the workload, using 10, 30, and 50 hours as maximum subscription times. The first chart (left) shows that the cloud can attend to all the users arriving during the first day, although some of them have to wait. However, in the following days, we have that many users must leave the cloud without being served. Using a maximum subscription time of 30 hours (central chart), users are served during the first 3 days. However, as the waiting time increases, they start to leave during the fourth day. Finally, considering 50 hours as maximum subscription time, the cloud can attend to all the users. Their waiting times reach up to 40.29 hours for some users, which is the inflection point in this case.

Figure 7 shows the results obtained for a cloud with 96 physical machines and a maximum subscription time of 2, 6, and 10 hours. The inflection point, in this case, is about 9.82 hours. The first chart of this figure (left) refers to the cloud processing the workload using a maximum subscription of 2. In this case, we observe again that many users leave the cloud from the second day onwards, because we still have in execution the applications from previous users, even from previous days. If we consider a maximum subscription time of 6 hours, only a few users leave the system (central figure) and taking 10 hours (right figure) all users are served, and the waiting times tend to stabilize in the peaks.

The results for a cloud configuration with 128 nodes are presented in Figure 8, using 2, 4, and 6 hours as maximum subscription times. In this case, the inflection point is about 5.20 hours. This system is now able to serve the users’ requests for the first two days, with small waiting times, and from the third day onwards the waiting times increase above 2 hours. Thus, we have reduced the inflection point in a factor close to 8 in comparison with the 64-nodes configuration, and in a factor of 2 with respect to the 96-nodes configuration.

4.3. Discussion of the results

This section provides a brief discussion of the obtained results. Table 1 shows the values for the inflection points in all the experiments.

Regarding the first set of experiments, where the studied clouds process a workload generated using a normal distribution, we observe that, in general, the maximum subscription time has a significant impact on the overall system performance. Increasing this parameter allows more users to be attended to by the system. Additionally, increasing the number of physical machines also impacts positively in the cloud performance, which allows us to reduce the maximum subscription time in order to process the same amount of users. In particular, a cloud with 64 physical machines is unable to attend to all the user’s requests when short subscription times are considered, and the users must set up subscription times of several days if they want their applications to be executed. In contrast, with a cloud with 128 nodes we have seen that the inflection point has been reduced by a factor of 12, and with the configuration with 96 nodes the reduction is about 1/3.

The results obtained in the next set of experiments, that is, where the clouds process a workload consisting of daily blurs, render similar results. A cloud with 96 nodes would offer responses below 10 hours for the users’ requests, and an investment to improve the cloud infrastructure up to 128

Table 1: Inflection points for the user waiting times (hours).

<table>
<thead>
<tr>
<th></th>
<th>64 nodes</th>
<th>96 nodes</th>
<th>128 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>63.02</td>
<td>21.32</td>
<td>5.16</td>
</tr>
<tr>
<td>Case 2</td>
<td>40.29</td>
<td>9.82</td>
<td>5.20</td>
</tr>
</tbody>
</table>
nodes would produce a gain of one half in the waiting times. Obviously, the final decision strongly depends on the applications and the users, who usually pay for the use of the cloud, so the cloud provider should take into account all of these aspects to make a final decision.

Broadly speaking, these results provide relevant and valuable information for the cloud provider, so as to improve the cloud configuration with the goal of increasing the overall income by adapting the physical resources and the internal configuration parameters.

5. Conclusions

In this paper, we have studied the impact of the users’ maximum subscription time in three different cloud configurations, considering an infrastructure consisting of 64, 96, and 128 physical machines, respectively. Two models of workload were analyzed, taking two different distribution functions for the users’ arrivals, namely, a normal and a cyclic normal. Thus, in the first case, we analyzed the impact of a single peak in the users’ arrivals, and in the second case, we considered daily peaks at midday. In this study, the responsiveness of the cloud was then analyzed, to conclude which configurations provide better results according to the workload submitted and the maximum subscription times indicated by the users. We concluded that increasing the number of physical machines and the maximum subscription time produce better responsiveness. However, the cloud provider must take the final decision, that is, to make an investment by including more resources to the cloud, or to reduce the overall cloud performance by increasing the maximum subscription time for the users.

As future work, we will extend this study by considering other parameters, such as the offered VMs, the storage system and the communication network. We will also consider some other distribution functions for the user arrivals, such as the exponential and Erlang distributions. Furthermore, we plan to include costs in the use of the cloud by the users, to make a deeper analysis of the profits.

ACKNOWLEDGEMENTS

This work was supported by the Spanish Ministry of Science and Innovation (co-financed by European Union FEDER funds) project references RTI2018-093608-B-C32 and RTI2018-095255-B-100. There was also support from the Junta de Comunidades de Castilla-La Mancha project SBPLY/17/180501/000276/01 (cofunded with FEDER funds, EU), the Region of Madrid (grant number FORTE-CM, S2018/TCS-4314), and the Madrid Government (Comunidad de Madrid-Spain) under the Multiannual Agreement with the Complutense University as part of the Program to Stimulate Research for Young Doctors in the context of the V PRICIT (Regional Programme of Research and Technological Innovation) under grant PR65/19-22452.

References

An Analysis of the State of the Art of Machine Learning for Risk Assessment in Software Projects

André Sousa¹, João Pascoal Faria², João Mendes-Moreira²
¹Master’s in Software Engineering, ²Department of Informatics Engineering
Faculty of Engineering of the University of Porto
Porto, Portugal
{up201902618, jpf, jmoreira}@fe.up.pt

Abstract—Risk management is one of the ten knowledge areas discussed in the Project Management Body of Knowledge (PMBOK), which serves as a guide that should be followed to increase the chances of project success. The popularity of research regarding the application of risk management in software projects has been consistently growing in recent years, particularly with the application of machine learning techniques to help identify risk levels or risk factors of a project before the project development begins, with the intent of improving the likelihood of success of software projects.

This paper provides an overview of various concepts related to risk and risk management in software projects, including traditional techniques used to identify and control risks in software projects, as well as machine learning techniques and methods which have been applied to provide better estimates and classification of the risk levels and risk factors that can be encountered during the development of a software project. The paper also presents an analysis of machine learning oriented risk management studies and experiments found in the literature as a way of identifying the type of inputs and outputs, as well as frequent algorithms used in this research area.

Index Terms—Risk Management, Risk Assessment, Software Projects, Machine Learning, Classification

I. INTRODUCTION

According to the Project Management Body of Knowledge, a project risk is “an uncertain event which, if it occurs, has a positive or negative effect on one or more project objectives” [1]. Software projects are notoriously complex development activities, and thus the concept of risk cannot be ignored when considering this type of projects.

In 2015, the Standish Group International’s CHAOS Report [2], a study of the success of software projects, reported a 29% success rate for the roughly 5000 projects investigated. A project is considered successful if it is completed within its allocated budget, original delivery deadline, and with all of the features that were planned at the start of its development life cycle [3]. It also reported a 19% failure rate for the set of projects investigated, meaning the projects suffered from cost overruns, or time overruns, or lacked content that was initially specified.

However, it is in the category of challenged projects that we can find the largest percentage of projects. From 2011 to 2015, 49%, 56%, 50%, 55%, and 52% of the software projects, respectively, were considered challenged, meaning they were completed but either over-budget, over the allocated time estimates, or offering fewer features than originally planned. The consistently high percentage of challenged projects indicates that there is room for improvement in the success rate of a large amount of these projects. This is where the concepts of risk and risk management are important to consider.

Risk management is a process used for early identification, analysis, planning, and control of risks in a project [4], with the goal of minimizing negative risks and maximizing positive risks [5], also referred to as opportunities. By identifying the risks and creating mitigation plans to deal with them if they occur before the project development starts rather than coming up with strategies to deal with risks in the moment they materialize, project managers and development teams are better prepared to handle risks and their effects on a project, which in turn can lead to more projects being completed on time and within their allocated budgets. However, it is typically the first activity to be removed from the project management activities when a project falls behind schedule [6].

In recent years, there has been an increasing use of machine learning algorithms and techniques for risk assessment, particularly supervised learning ones where the model is trained using a data set, and then the same model is used to predict information on a new set of data (in this area, it could be to predict possible risk factors of a project based on its characteristics, such as team members, time, and allocated budget). Commonly used algorithms for supervised learning include Decision Trees, Naïve Bayes classifiers (NB), Neural Networks (NN), and Support Vector Machines (SVM).

The purpose of this paper is to provide an overview of the state-of-the-art in topics related to risk and risk management with regards to their application in the management of software projects, including traditional processes as well as the growing application of machine learning techniques to tackle the problems associated with risk assessment in software projects. The remaining sections of the paper are structured as follows. An overview of the concepts of risk and risk management in software projects, and different types of risks and risk management processes are presented in section 2. In section 3, a literature review of the application of machine learning techniques for risk management is performed, showing examples of their use in the prediction of risk levels of software projects. Lastly, section 4 concludes the paper.
II. Risk and Risk Management in Software Projects

Risk in software projects can be seen as “the potential that a chosen action or activity will lead to a loss or an undesirable outcome” [9]: “a set of factors or conditions that can pose a serious threat to the successful completion of a software project” [7]; or “the probability and impact of an event on a project” [8]. From these definitions, it is possible to identify some common themes, such as a risk possibly leading to a loss. In a software project, a loss can manifest itself through lower quality of the final product, increased costs, changes to the release date of the product, or, in a worst-case scenario, failure and cancellation [9].

Software projects can be impacted by various types of risks [10]:

- **Technical risks** - problems with the programming languages and frameworks of choice, project size, or processes. This type of risk can occur as a result of lack of experience or lack of maturity of the technologies used.
- **Management risks** - these risks can occur due to problems in communication with top management and customers, lack of planning, or lack of project management experience.
- **Financial risks** - problems regarding budget, cash flow, or doubts about the return on investment of the project.
- **Contractual and legal risks** - problems regarding adjusting the schedule or the requirements to fit the market, government regulations, or health and safety problems.
- **Personnel risks** - these can be due to conflicts among staff, ethical and moral issues, or productivity issues resulting from a combination of the aforementioned risks.
- **Other resource risks** - these occur due to situations that are generally not a responsibility of the project team, such as unavailability of computer resources or equipment.

When it comes to the most frequent specific risks in software projects, Boehm, regarded by many as one of the most important authors in this research area, listed risks such as personnel shortfalls, unrealistic schedules and budgets, and developing the wrong functions and/or user interface as some of the most frequent risks that have a direct effect on the success of software projects [8].

The high percentage of challenged projects seen in the Standish Group International’s CHAOS reports [2] is consistent with the information presented by Boehm [8], as the first and second most frequent risk items listed (personnel shortfalls and unrealistic schedules and budgets) are directly related to the concept of challenged projects, and often come as a consequence of the majority of the remaining risk items listed occurring during the development of a project.

To reduce the high percentage of challenged projects in the software industry, project managers must consider a wide variety of knowledge areas in order to manage their projects towards successful completion. One of those areas is risk management, as risks can be identified in various areas of a software project. In a software development project, risks can be influenced by the business domain, the business style, culture of the organization, and characteristics of the members involved in the project [11], so it is important to identify risks according to the environment in which the project is being developed. To facilitate this process, risk factor and item classifications found in the literature can be used. These classifications usually list the most frequent risk items that can affect a project’s path towards success, and teams can use these to evaluate if there is a possibility of any of those risks occurring during the development of their own projects, and if so, what could be their impact on the project. Essentially, those are the 2 parts that make up a risk: the likelihood of the risk happening, and the degree of impact it has on the project if it does occur.

Wallace’s categorization [7] of risk items according to six risk dimensions (Team, Organizational Environment, Requirements, Planning and Control, User, and Complexity) is still widely used in this research area. Not only does it present common risk items in software projects, but by grouping them according to a specific dimension within the areas that project managers have to consider in the development of a software project, it makes it so they can identify what areas are more likely to be problematic throughout the course of the development of the project and prepare their risk management strategies accordingly.

Those are just some examples of risk classifications that can negatively affect specific areas during the development of a software project. In reality, there are a lot more risks that can be identified, and doing it at an early stage of the project (ideally before development starts) is crucial for a successful development life cycle, as it means the project manager as well as the development team can start to plan actions to take if these risks materialize during the project development.

A good way of classifying the identified risks according to their priority is through the use of a portfolio chart, such as the one presented by Dr. Ernest Wallmüller [12], which can be seen in Table I.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>B A A</td>
</tr>
<tr>
<td>Medium</td>
<td>C B A</td>
</tr>
<tr>
<td>Low</td>
<td>C C B</td>
</tr>
</tbody>
</table>

Identifying, classifying, and prioritizing actions for risks according to their priority are just some of the phases of a process called risk management, which is defined by Standard ISO/IEC/IEEE 24765:2010 as “an organized process for” identifying and handling risk factors; assessing and quantifying the identified risks; and developing plans to deal with the identified risks [13].

In a very simplified way, the goal of risk management is to increase the probability of positive events on a software project, while at the same time decreasing the probability of negative events on the same project [4]. To do that, risk...
management is often divided into two key activities: risk assessment and risk control, which are composed of more specific steps.

In [8], Boehm split risk assessment into three phases - identification, analysis, and prioritization - and risk control into three other phases: management planning, resolution (or mitigation), and monitoring. Other researchers may change the categories to which these steps belong to, such as in [14], where risk assessment is made up of phases for identification, analysis, prioritization, planning, and resolution, and risk control is made up of only one phase, which is monitoring, but the key concepts and phases remain the same.

Risk assessment begins with the risk identification phase, where a list of risk items related to the project that have a higher chance of affecting the success of the project is created. Common techniques in this phase include checklists and decision-driven analysis. Afterwards, the loss probability and impact of each identified risk item is assessed in the risk analysis phase. Analysis of factors such as quality, reliability, and availability is a common task in this phase. However, there is usually some uncertainty when it comes to estimating the losses that are a result of the occurrence of a risk [8], so the assessments done are very subjective, and often the result of interviewing domain experts [8]. Next, in the risk prioritization phase, the identified risks are ordered through the use of techniques such as the analysis of risk exposure and risk reduction.

With the first three phases of risk management (according to Boehm) completed, the risk control activities can begin with the risk management planning step, which addresses the risk items identified through processes such as buying information (e.g., investing in a prototype to better understand the specific risk), risk avoidance, risk reduction, risk transfer, and risk plan integration. Common techniques used in this step are checklists of risk-resolution techniques, cost-benefit analysis, and risk management plan outlines. Afterwards, in the risk resolution phase, the identified risk items are analyzed and decisions are taken regarding what action to take against the risks in order to mitigate them. Boehm identified several fundamental risk mitigation strategies, such as understanding the risk or removing the risk from the project’s critical path [4].

Lastly, in the risk monitoring phase, the project’s progress is tracked towards completion by resolving the previously identified risk items and taking corrective action whenever necessary through the use of techniques such as milestone tracking and risk assessment.

Boehm’s risk management model is frequently referenced in the literature, but there are several other traditional risk management models and processes that can be found in the literature. The authors in [15] analyzed several risk management models and processes, such as:

- **Team Risk Management (TRM) [16]** - risks are managed in the full software development life cycle, and all members and stakeholders are involved, improving the efficiency of the decision-making process. TRM frequently ensures continuous risk management through regular reviews and monitoring of the implemented processes.

- **Softrisk management technique [17]** - this technique is constructed on the basis of documentation and gives special focus to extreme risks by focusing on what can be leading to those risks. Re-estimation, re-prioritization, reassessment, and re-documentation are performed to also guarantee continuous risk management.

- **Wallmüller’s Risk Management Process [12]** - risk management activities are conducted by the project team at the same points where the cost, time, quality, and requirement management activities are performed. A major point of difference compared to the previous models is the introduction of risk management roles which are assigned to different members of the team, thus making sure the entire team contributes to the risk management tasks and is up-to-date on the status of risks in the project.

There is another area that has been gaining a tremendous amount of attention, particularly in recent years, with the goal of improving risk management processes in software projects, and that is the application of machine learning techniques and methods to improve the risk management workflow in software development companies.

### III. Literature Review of Machine Learning Approaches in Risk Management

Machine learning in risk management has obtained increasing popularity in recent years, and a lot of different approaches have been used. For the purposes of the analysis of the state of the art performed in this paper, the focus was on finding practical applications of machine learning to predict possible project risks or an overall risk level of a project. From there, it was possible to identify not only some of the most frequently used algorithms and evaluation metrics, but also the type of information used as inputs used to train the models.

Throughout the creation of this paper, bibliographic databases such as Scopus and DBLP were used to search for various articles, scientific papers, and surveys related to this topic. Searches were performed using keywords such as “risk assessment”, “machine learning”, and lastly “software projects” to reduce the scope of the results to the application of machine learning for risk assessment specifically in the software development industry. By reading the abstracts and briefly analysing the contents of the search results, the ones that were considered more relevant were read and analysed in more detail. Some examples of studies and experiments done in this research area are described below, and can be seen in greater detail in terms of inputs and outputs used in table II.

In [18], an Artificial Neural Network model was created to predict deviations in new software projects. The inputs to the model were the risk factors detected in the projects, and the outputs were the differences found in time, budget, and number of personnel, number of completed work packages, and success of the project under investigation. This experiment showed the applicability of Neural Networks when the intended information spans more than one category (in this
case, the deviations in five attributes related to the project), as well as the fact that the model can have a great performance and accuracy, as seen in its results.

A Neural Network model was also created in [19], together with a Support Vector Machine model to compare both approaches and their accuracy in evaluating the risk level of software projects. The input used was a vector of risk factors of 120 software projects, collected after several interviews with experts in the industry, which were then grouped according to six different risk categories (Environment Complexity, Project Requirement Complexity, Cooperation, Team, Project Management, and Engineering). The output was the predicted outcome of the project (“successful”, “failed”, or “challenged”). The Support Vector Machine model had a higher accuracy compared to the Neural Network method (80% vs 70%, respectively) due to NN’s tendency in finding a local optima [19], but after changes were made to the NN method by optimizing it with a Genetic Algorithm (GA), this made it so the NN-GA method surpassed SVM in accuracy (85% vs 80%, respectively) by reducing the search for a local optima.

In [20], the author proposes a Neural Network architecture with a back propagation algorithm to learn the patterns of a data set of projects completed in the past, which also includes 22 project risk factors of areas such as estimations, requirements (e.g., frequent changes to requirements), and team organization (e.g., lack of skills or experience). The output of the model was a classification of the risk level of the project: “runaway” or “success” projects through the use of a Naive Bayesian classifier. These characteristics are classified according to five different categories: requirements (e.g., ambiguity of requirements), estimations (e.g., lack of stakeholders present for estimation process), planning (e.g., unspecified milestones), team organization (e.g., lack of skills or experience), and project management (e.g., inadequate project monitoring). 10-fold cross validation was used to evaluate the effectiveness of their solution, showing a predictive accuracy of 82.5%, with 33 out of 40 projects classified correctly.

The authors in [11] developed an approach to predict runaway projects (projects that greatly exceed budget and deadlines and have failed to produce an acceptable deliverable) in an organization through the use of a questionnaire to identify the characteristics of projects, and then classify them into “runaway” or “success” projects through the use of a Naive Bayes classifier. These characteristics are classified according to five different categories: requirements (e.g., ambiguity of requirements), estimations (e.g., lack of stakeholders present for estimation process), planning (e.g., unspecified milestones), team organization (e.g., lack of skills or experience), and project management (e.g., inadequate project monitoring). The authors found that it had an accuracy of 1% to 7% higher than the other models tested (Logistic Regression, Decision Tree, and Naive Bayes), which they attributed to the incorporation of expert domain knowledge and causality discovery into the BBN.

In [23], the authors used a Support Vector Machine to model risk classification in software projects. The model classified projects as either high risk or low risk. SVM was also used in [24] to predict the risk level of different projects as either “low”, “medium”, or “high”. A Neural Network was used for comparison, and the authors found that the SVM was more accurate (85% accuracy of SVM compared to 75% of the NN).

Multiple Logistic Regression was used in [25] to classify different characteristics of software projects as either a “risk” or a “non-risk”. The input data was obtained through questionnaires sent to experts in the software project development and management fields, which asked them to classify risk factors from 8 categories (User, Requirements, Estimations, Cost, Schedule, Planning and Control, Team, Software) according to their risk level on a scale from 1 to 5.

Lastly, the authors in [26] used Logistic Regression to classify projects as either “risk” or “not risk”. Responses to a questionnaire focusing on 5 viewpoints of key risk factors (Requirements, Estimations, Planning, Organization, Management) were used as the input data, and the model developed classified 35 out of 40 projects correctly.

As can be seen, there are a lot of possibilities when it comes to machine learning models that can be used to predict risks in software projects. However, there are definitely areas in this field that can be explored further in order to improve the applicability of machine learning models for risk assessment.

Some of the papers presented in table II compare different machine learning algorithms (e.g., [19] and [24]) with the goal of comparing their predictive performance in the context of a specific problem. However, a greater focus should be placed in also comparing them in terms of interpretability and the performance trade-offs involved in more interpretable algorithms.

Interpretability in machine learning is defined as “the degree to which a human can understand the cause of a decision” [27]. Interpretable machine learning models make it easier to understand not only the prediction made by the model, but more importantly why that prediction was made. If a prediction does not match what was initially expected, developers can use this information to identify possible issues in the data set, the model, or possibly both. However, there is a trade-off involved with interpretable machine learning algorithms, namely the fact that predictive performance tends to be lower with these algorithms.

Additionally, considering the popularity of project management software such as JIRA and Asana, one of the next areas of focus in this research field could be the creation of machine learning models that can be integrated with these tools. This integration with tools used for daily project management tasks could make it so risk management becomes just another step in the project management cycle, rather than a process which requires a large overhaul in an organization’s workflow in order to integrate it in their processes.


<table>
<thead>
<tr>
<th>Reference</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Algorithm(s)</th>
<th>Evaluation metric(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Novel Model for Risk Estimation in Software Projects using Artificial Neural Network [18]</td>
<td>45 risk factors of 20 software projects (70% of data used for training, 30% for testing)</td>
<td>Deviations in project duration, cost, number of personnel, completed work packages, project success</td>
<td>Neural Network</td>
<td>Training $R^2 = 0.9978$ Testing $R = 0.9935$ Validation $R = 0.996$ MSE$^2 = 0.001$</td>
</tr>
<tr>
<td>Software Project Risk Management Modelling with Neural Network and Support Vector Machine Approaches [19]</td>
<td>Data of 120 software projects collected through questionnaires distributed in cities in China (83.3% of data used for training, 16.7% for testing)</td>
<td>Classification of projects as either “successful”, “challenged”, or “failed”</td>
<td>Neural Network</td>
<td>Accuracy = 98%</td>
</tr>
<tr>
<td>Discriminating Risky Software Project Using Neural Networks [20]</td>
<td>22 attributes of 40 projects in the OMRON database (80% of data used for training, 20% for testing)</td>
<td>Risk level of the project - “risky” or “not risky”</td>
<td>Neural Network</td>
<td>Accuracy = 80%</td>
</tr>
<tr>
<td>An Empirical Evaluation of Predicting Runaway Software Projects Using Bayesian Classification [11]</td>
<td>Responses on a 4 point Likert scale to a questionnaire focusing on 5 viewpoints of key risk factors in 40 SSBC projects (10-fold cross-validation used for testing)</td>
<td>Project classification as either “runaway” or “success”</td>
<td>Bayesian classifiers</td>
<td>Accuracy = 82.5%</td>
</tr>
<tr>
<td>A Probabilistic Software Risk Assessment and Estimation Model for Software Projects [21]</td>
<td>Assessment of 27 risk factors (low, medium or high) in 12 software projects</td>
<td>Probability of the project being of low, medium, or high risk</td>
<td>Bayesian classifiers</td>
<td>MMRE$^5 = 0.03842$ BMMRE$^6 = 0.03911$</td>
</tr>
<tr>
<td>Software Project Risk Analysis using Bayesian Networks with Causality Constraints [22]</td>
<td>Software project data from 302 projects collected through questionnaires (10-fold cross-validation used for testing)</td>
<td>Classification of project’s performance based on risks identified as “low” or “high”</td>
<td>Bayesian network with causality constraints</td>
<td>Accuracy = 75.15%</td>
</tr>
<tr>
<td>Classification of Risk in Software Development Projects using Support Vector Machine [23]</td>
<td>530 samples of a data set created from information of software development projects (70% of data used for training and 30% for testing)</td>
<td>Project risk classification as either “low risk” or “high risk”</td>
<td>Support Vector Machine</td>
<td>Accuracy = 99.51% AUC$^7 = 98%$</td>
</tr>
<tr>
<td>An Intelligent Model for Software Project Risk Prediction [24]</td>
<td>64 risk factors of data from 120 projects (83.3% of data used for training, 16.7% used for testing)</td>
<td>Classification of projects as either “successful”, “challenged”, or “failure”</td>
<td>Neural Network</td>
<td>Accuracy = 75%</td>
</tr>
<tr>
<td>Prediction of Risk Factors of Software Development Project by Using Multiple Logistic Regression [25]</td>
<td>Data obtained from questionnaires regarding the risk level of 70 software projects</td>
<td>Classification of characteristics of a software project as “risk” or “non risk”</td>
<td>Multiple Logistic Regression</td>
<td>Accuracy = 90%</td>
</tr>
<tr>
<td>An Empirical Approach to Characterizing Risky Software Projects Based on Logistic Regression Analysis [26]</td>
<td>Responses on a 4 point Likert scale to a questionnaire focusing on 5 viewpoints of key risk factors in 40 SSBC projects</td>
<td>Classification of projects as either “risky” or “not risky”</td>
<td>Logistic Regression</td>
<td>Accuracy = 87.5%</td>
</tr>
</tbody>
</table>

1 Defined by the authors as Regression value, indicating the correlation between the predicted values and the observed values. A higher value indicates better results.
2 Mean Squared Error. A smaller value indicates better results.
3 True Positive Rate - the percentage of positive cases that were correctly identified. A higher value indicates better results.
4 True Negative Rate - the percentage of negative cases that were correctly classified. A higher value indicates better results.
5 Mean Magnitude of Relative Error. A lower value indicates better predictive performance.
6 Balanced Mean Magnitude of Relative Error. Due to the fact that MMRE penalizes overestimates more than underestimates, a balanced MMRE is also used in this experiment. As with MMRE, a lower value indicates better predictive performance.
7 Area Under the ROC Curve - the probability of the model ranking a random positive example higher than a random negative one. AUC returns a value between 0 and 1, where the higher the AUC, the better the model is at distinguishing positive and negative classes.
Lastly, the use of accuracy as the sole evaluation metric to assess the quality of the models developed (e.g., [19], [25], [26]) is sometimes not enough. The usefulness of accuracy as an evaluation metric depends on the balance of the classes in the problem at hand. As an example, if there are three possible risk levels (e.g., low, medium, and high) to choose from in the data set’s dependent variable, and 70% of the samples are of class “low”, 10% are of class “medium”, and the remaining 20% are of class “high”, the model can easily obtain a high training accuracy by just predicting the majority of the testing samples to be of class “low”. In classification problems, additional metrics such as AUC, True Positive Rate, and True Negative Rate should also be closely looked at to determine if the model is truly returning good results, or if it is only achieving a high accuracy by predicting the testing samples as being of the majority class most of the time.

IV. CONCLUSION

As software projects can face a lot of different problems before they are released to the market, it is important to at least identify possible risks that can occur before development starts, making it possible to start planning risk management and mitigation strategies if the risks materialize, rather than dealing with the problems as they appear. Risk management in software projects is a research area with consistently growing popularity, especially when combined with machine learning approaches to create models that can identify or predict risks before project development starts, with the goal of identifying risks in a software project, and ultimately develop and implement strategies to prevent or limit the impact of the identified risks if they materialize during the project’s development.

Explainable AI is also a research field with increasing research that should be considered to explain the prediction of black-box models, such as Neural Networks or Support Vector Machines. One of the next steps in this research area should be to focus on understanding the predictions that are made by the models by using interpretable models or black-box models but, in the latter case, with their predictions explained by explainable AI. Which would be the best?

Lastly, the creation of machine learning models in software packages that can then be integrated with popular project management tools such as JIRA or Asana should be analysed more closely.

REFERENCES

Towards a Comprehensive Understanding of Agile Teamwork: A literature-based Thematic Network


* Intelligent Software Engineering (ISE) Group @ VIRTUS, Federal University of Campina Grande
Emails: {arthurfreire, manuel}@copin.ufcg.edu.br
and {mirko, alexandre.costa, kyller, hyggo, perkusic}@virtus.ufcg.edu.br

Abstract— Agile Software Development (ASD) has become the mainstream software development method of choice. Its core fundamentals are based on Teamwork factors and the higher value it gives to individuals and their interactions over processes and tools. Teamwork and human factors have been addressed as essential topics in the literature, and researchers have stated the importance of measuring it to increase the chances of success of ASD projects. However, there is no common understanding regarding the factors that should be considered for defining an ASD Teamwork construct. Driven by this problem, this paper presents a thematic network that defines the themes (i.e., factors) that should be considered when addressing ASD Teamwork. The ASD Teamwork thematic network is the result of a process that consisted of (i) defining the studies used as a data source through a literature review; (ii) extracting data from these studies; (iii) translating this data into codes; (iv) translating the codes into themes; (v) creating the model of higher-order themes; and, (vi) assessing the trustworthiness of the synthesis. The resulting thematic network comprises four higher-level themes: Cohesion, Orientation, Shared Leadership, and Autonomy. We believe that the constructed thematic network can be generalized to ASD and used as the basis by researchers who intend to explore ASD Teamwork. Further, practitioners can use our results to understand agile teams’ dynamics better and improve their efficiency.

Index Terms—Teamwork; Agile Software Development; Agile; Thematic Network; Construct

I. INTRODUCTION

According to Hoda et al. [1], Agile Software Development (ASD) has become the mainstream development method of choice. The main reason for adopting ASD is its capability to respond to environmental changes, such as requirements, quickly. Usually, agile initiatives embrace iterative development, which means dividing the delivery process into short iterations, allowing requirements to be refined on a regular basis [2].

Additionally, to enable responsiveness to change, the Agile Manifesto [3] states that ASD values individuals and interactions more than processes and tools [4]. The team’s importance in ASD is evidenced by the Agile Manifesto having six out of the twelve principles directly related to the team (i.e., individuals). Moreover, other researchers have recently assessed the relationship of team members’ personality on Teamwork quality (TWQ) in the context of ASD [5].

Research has shown that TWQ has a positive impact on team development [6], [7], and is, consequently, essential for succeeding with ASD [8], [9], [10], [11], [12]. Batista et al. [13] discussed that the effective combination of individual parts, often carried out by software development teams, requires interactions among team members and the coordination of interdependent individual and team level tasks. Given its impact, researchers argued about the importance of assessing TWQ to increase the chances of succeeding with ASD [13], [14].

In this context, researchers have proposed instruments for assessing ASD Teamwork quality. Moe et al. [15] presented a Radar Plot that considers five dimensions for assessing TWQ: Shared Leadership, Orientation, Redundancy, Learning, and Autonomy. Lindsjørn et al. [16] presented a Structural Equation Model, based on a differentiated replication [17] from [6], which considered that the Teamwork construct is comprised of six variables: Communication, Coordination, Balance of Member Contribution, Mutual Support, Effort, and Cohesion. Finally, Freire et al. [14] proposed a Bayesian networks-based model with 16 variables, which had its practical utility positively assessed in a case study.

Unfortunately, there is still no common understanding of what factors should be considered in the ASD Teamwork construct. Except for the instrument proposed by Freire et al. [14], which claimed to have considered the factors presented in Lindsjørn’s et al. [16] and Moe’s et al. [15] instruments, there is no direct similarity between any of the ASD Teamwork factors in these last two instruments. Even though Freire et al. [14] considered factors from the other two instruments, it does not contain them all; it does not include the variable Balance of Member Contribution, which is part of the ASD Teamwork construct presented in [16]. Moreover, except from the ASD Teamwork factors presented in [14], we did not find other study in the literature that presented a similar list based on the literature. Therefore, we believe that there is still a need to develop a comprehensive and shared understanding of the essential factors included in the ASD Teamwork construct.

To address this gap, we identified the ASD Teamwork factors presented in the literature through thematic analysis, following the guidelines presented by Cruzes and Dyba [18]. This paper presents the employed methodology and the resulting thematic network, which comprises the factors (i.e.,
themes) for ASD Teamwork and their relationships.

The remainder of this paper is organized as follows. Section II presents necessary background and related work. Section III presents the employed methodology. Section IV describes the resulting thematic network. Section V discusses this study’s findings in light of its implications for research and practice. Section VI discusses this study’s threats to validity. Finally, Section VII presents our final remarks.

II. BACKGROUND AND RELATED WORKS

Freire et al. [14] present a Bayesian networks-based model to assess and improve the TWQ in the ASD context. To build the model, the authors listed many ASD Teamwork key factors extracted from the literature. Based on the knowledge of an expert and the resulting list, they used reasoning on a top-down approach - starting with the target node (i.e., Teamwork quality) - breaking down higher-level factors into others they judged more observable. In [14], the authors also present the results of a case study in which their instrument’s practical utility was assessed. They concluded that their model helps agile teams assess TWQ and identify improvement opportunities, is easy to learn, and the cost-benefit for using it with the proposed procedure is positive.

Lindsjørn et al. [16] presented a Structural Equation Model instrument, based on a differentiated replication [17] from a study of Hoegl and Gemuenden [6], which considered that the Teamwork construct is comprised of six variables: Communication, Coordination, Balance of Member Contribution, Mutual Support, Effort, and Cohesion. Lindsjørn et al. assumed that study presented in [6] focused on traditional software development methodologies and analyzed how the theory presented in it applies to ASD. As a result, they concluded that the quality of the Teamwork is a major factor in improving team performance, especially for the product’s quality.

Moe et al. [15] propose a Radar plot-based instrument to help diagnose agile Teamwork, which considers five dimensions: Shared Leadership, Orientation, Redundancy, Learning, and Autonomy. The instrument was presented to a group of experts comprised of 35 people. They found the model useful for understanding team problems, such as the team agreeing on using test-driven development. According to the authors, the instrument gives researchers and developers a common language for discussing Teamwork.

III. RESEARCH METHODOLOGY

This section presents the employed research methodology and partial results - due to space limitations - of the steps executed for constructing the proposed thematic network. This study aimed to understand the factors (or dimensions) of agile Teamwork. Given this, we defined the following Research Question (RQ):

**RQ:** Which themes should be considered when defining ASD Teamwork?

To answer RQ, we employed a thematic analysis approach [18]. The employed process can be divided into two main steps: (i) Data Source Definition and (ii) Thematic Network Construction. The output of step (i) was used as input for the employed technique we applied in step (ii). In what follows, Section III-A describes step (i) and Section III-B, step (ii).

A. Data Source Definition

The first step was to define our data source, in other words, identify studies that present ASD Teamwork factors. For this purpose, we used as the initial set of studies the fifteen studies pointed by Freire et al. [14] (i.e., seed set). Later, we employed a Forward Snowballing [19] to identify additional primary studies. To select the studies to be considered on the Forward Snowballing, we employed the following inclusion criteria:

1. Published in 2019 or later
2. Written in English
3. Published in a Conference proceedings or Journal as full papers
4. Focused on industry context (i.e., not considering papers/studies with students or in the academic context)
5. Presents ASD Teamwork factors

We decided to define stringent criteria to consider a paper relevant in the Forward Snowballing step because we assumed our start set to be reliable. Such reasoning follows from it being based on a previews literature review discussed in Freire et al. [14]. Further, our goal was not to quantify the frequency of appearance of a factor in the literature in favor of its relevance, but to identify high-quality studies that discussed relevant agile Teamwork factors.

We managed the Forward Snowballing process in an online Google Spreadsheet (see Appendix Table I)\(^1\). For each paper in the seed set, we used Google Scholar to identify papers that cited it. Wohlin [19] recommends Google Scholar to avoid bias in favor of any specific publisher. We filtered the results given the publication date, including only those published from 2019 and later (i.e., we applied Inclusion criteria #1). For the resulting papers, we screened them and applied Inclusion criteria #2, #3, and #4. Finally, we analyzed the resulting papers’ title and abstract in light of Inclusion criteria #5.

On the first iteration, for the 15 papers in the seed set, we found 13 other papers, from which only six were considered relevant. The remaining seven were discarded for the following reasons: being book chapters or workshop papers, did not present ASD Teamwork factors, or were related to agile transition only. Given this, our data source for constructing the thematic network contained 21 papers.

B. Thematic Network Construction

To build the thematic network, we applied the thematic analysis process proposed by Cruzes and Dyba [18]. Their guideline is comprised of five steps: (i) data extraction; (ii) code data; (iii) translate codes into themes; (iv) create a model of higher-order themes; and (v) assess the trustworthiness of

---

\(^1\)https://doi.org/10.6084/m9.figshare.14214431.v1
the synthesis. All steps in this process were performed by the first author and checked by the other authors.

For the data extraction step, step (i), we extracted the Teamwork factors from the papers in our study’s data source (see Section [14]), alongside text segments describing or explaining them. The output was a list of 74 ASD Teamwork factors that served as input for the next step. In step (ii), we coded the 74 ASD Teamwork factors’ text segments identified in step (i) given their naming or description. For example, we labeled the factors Adaptability, Learning, Learning, and Team Adaptation into the code Learning in our further steps; while the factors Team Leadership, Shared Leadership, Leadership and Shared Responsibility were labelled with the code Shared Leadership.

Then, in step (iii), we translated the codes into themes. This step has some similarities with the previous one, given the mechanism that we adopted: group one or more codes into a theme representing them all. The themes comprised their representation into a specific Teamwork area that the authors judged valid given the codes’ and factors’ description. We grouped the codes that semantically represented a given theme and considered the ones that we judged essential to the theme (i.e., comprised essential attributes for the theme concept). For instance, we translated the codes Coordination, Performance Monitoring, Task Novelty, and Familiarity into Coordination. In this case, we considered that Coordination and Performance Monitoring have similar definitions (i.e., semantics) with regards to the higher-order theme (i.e., Coordination). For Task Novelty, according to Marsicano et al. [20], when its value is low, it is more likely that team members assign work between them adequately. This concept is also related to the concept of Familiarity, and according to the previous description of Task Novelty, we interpreted that they were also related to Coordination (i.e., task assignment).

After identifying 13 themes, we found that describing the ASD Teamwork in light of them was complex. Thus, we executed step (iv) by refining them into higher-order themes. We started by relating the 13 themes into a higher-order theme for the overall ASD Teamwork, resulting in a tree scheme, with ASD Teamwork theme as the root node and the remaining 13 themes as leaf nodes.

Then, we identified new middle-level themes by grouping the leaf themes given their names and related factors. For example, we considered Cohesion a higher-order theme that comprises both Communication and Personality. The reasoning behind this is because Cohesion, as described in its related codes and factors, is directly related to the interpersonal attraction of team members and their willingness to continue working together. Based on this, we considered that the exchange of information between team members (i.e., Communication) and the mixture of personalities (i.e., Personality) contribute to it.

Notice that our goal was not to define a cause-consequence model, considering, for instance, the temporal relationship between the themes and codes, but to simplify the comprehension of ASD Teamwork dimensions. For example, even though Communication could be related to other themes, we associated it with the theme with the closest definition.

To perform step (v) and validate the synthesis’s trustworthiness, we reviewed the process we adopted regarding reducing the bias of the researchers by relying on the description of codes and factors that comprise the themes based on the peer-review adopted. Besides, we believe that the themes are consistent and understandable.

IV. RESULTS

This section presents the identified ASD Teamwork factors (see Table I) and the resulting thematic network (see Figure 1). In Table I, each line presents information for an identified code (i.e., ASD Teamwork factor). For each code, it presents the relative (i.e., “%” column) and absolute frequency (i.e., “Freq” column) of the code’s appearance on the seed set. Further, for improving understandability, column “Distribution” presents a visual representation of the code’s frequency of appearance. Finally, column “Themes” displays the theme mapped for each code. The complete information on the thematic network process is available in the Appendix.

From this point forward, we explain the reasoning behind the relations that we have defined for the themes presented in Figure 1. The resulting thematic network is composed of four higher-level themes: Cohesion, Orientation, Shared Leadership, and Autonomy. As previously discussed in Section III-B, the relationship between Cohesion with Communication and Personality follows from the reasoning that both concepts are part of Cohesion.

Orientation refers to the team members’ belief in the team goals’ importance over individual members’ goals and their propensity to take others’ behavior into account during group interaction. We considered as sub-themes for this theme: Feedback, Coordination, Collaboration, and Learning. By analyzing these themes’ concepts, we noticed that having the capabilities to coordinate the work among team members (i.e., Coordination) in a collaborative environment (i.e., Collaboration) that leverages constant feedback (i.e., Feedback) between the team members seemed strongly related to keeping team goals a top priority (i.e., Orientation).

Learning refers to the ability to understand and recognize deviations and readjust accordingly. Moreover, according to Ringstad et al. [11], it is also related to the development of shared mental models. Hence, we judged these capabilities as fundamental to keeping a good team Learning. Expertise is directly related to Learning because it comprises concepts such as Collective Knowledge, Redundancy, Adequate Skills, and Team Experience with Work, which we judged to be characteristics that influence the teams’ learning capabilities.

The third higher-level theme is Shared Leadership, which relates to the provisioning of direction, structure, and support for the team members, with the responsibility and decision authority for doing so being shared between the team members. We judged that having good Management Mechanisms, which includes planning, discussing, implementing the required changes, and evaluating the success and the taken
<table>
<thead>
<tr>
<th>Codes</th>
<th>%</th>
<th>Freq</th>
<th>Distribution</th>
<th>Themes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination</td>
<td>23.8%</td>
<td>5</td>
<td>[XXXXX]</td>
<td>Coordination</td>
</tr>
<tr>
<td>Performance Monitoring</td>
<td>42.85%</td>
<td>9</td>
<td>[xxxxxxxxxx]</td>
<td></td>
</tr>
<tr>
<td>Task Novelty</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Familiarity</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Culture</td>
<td>19%</td>
<td>4</td>
<td>[XXXX]</td>
<td>Organization Culture</td>
</tr>
<tr>
<td>Structure</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Team Size</td>
<td>9.5%</td>
<td>2</td>
<td>[XX]</td>
<td></td>
</tr>
<tr>
<td>Organization Support</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Individual Differences</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Members Personality</td>
</tr>
<tr>
<td>Personality</td>
<td>14.3%</td>
<td>3</td>
<td>[XXX]</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>19%</td>
<td>4</td>
<td>[XXXX]</td>
<td>Management Mechanisms</td>
</tr>
<tr>
<td>Planning</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Discussion</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Evaluation</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Information Radiators</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Decision Making</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Orientation</td>
<td>33.33%</td>
<td>7</td>
<td>[xxxxxxxxx]</td>
<td>Orientation</td>
</tr>
<tr>
<td>Value Diversity</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Goals</td>
<td>9.5%</td>
<td>2</td>
<td>[XX]</td>
<td>Expertise</td>
</tr>
<tr>
<td>Roles</td>
<td>9.5%</td>
<td>2</td>
<td>[XX]</td>
<td></td>
</tr>
<tr>
<td>Holistic Team Involvement</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Team Experience in the Organization</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Collaboration</td>
</tr>
<tr>
<td>Trust</td>
<td>23.8%</td>
<td>5</td>
<td>[XXXXX]</td>
<td></td>
</tr>
<tr>
<td>Motivation</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Shared Leadership</td>
</tr>
<tr>
<td>Norms</td>
<td>9.5%</td>
<td>2</td>
<td>[XX]</td>
<td></td>
</tr>
<tr>
<td>Tools knowledge</td>
<td>9.5%</td>
<td>2</td>
<td>[XX]</td>
<td></td>
</tr>
<tr>
<td>Collective Knowledge</td>
<td>19%</td>
<td>4</td>
<td>[XXXX]</td>
<td></td>
</tr>
<tr>
<td>Adequate Skills</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Redundancy</td>
<td>33.33%</td>
<td>7</td>
<td>[xxxxxxx]</td>
<td>Collaboration</td>
</tr>
<tr>
<td>Team Experience with Work</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Interdependence</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Shared Leadership</td>
</tr>
<tr>
<td>Collaboration</td>
<td>33.33%</td>
<td>7</td>
<td>[xxxxxxx]</td>
<td></td>
</tr>
<tr>
<td>Shared Leadership</td>
<td>38.1%</td>
<td>8</td>
<td>[xxxxxxxxxx]</td>
<td></td>
</tr>
<tr>
<td>Formal Leadership</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Autonomy</td>
</tr>
<tr>
<td>Autonomy</td>
<td>19%</td>
<td>4</td>
<td>[XXXX]</td>
<td></td>
</tr>
<tr>
<td>Task Control</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td>Feedback</td>
</tr>
<tr>
<td>Awareness</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Acceptance</td>
<td>4.75%</td>
<td>1</td>
<td>[X]</td>
<td></td>
</tr>
<tr>
<td>Feedback</td>
<td>14.3%</td>
<td>3</td>
<td>[XXX]</td>
<td>Learning</td>
</tr>
<tr>
<td>Learning</td>
<td>38.1%</td>
<td>8</td>
<td>[xxxxxxx]</td>
<td></td>
</tr>
<tr>
<td>Communication</td>
<td>42.85%</td>
<td>9</td>
<td>[xxxxxxxxxx]</td>
<td>Communication</td>
</tr>
<tr>
<td>Cohesion</td>
<td>14.3%</td>
<td>3</td>
<td>[XXX]</td>
<td>Cohesion</td>
</tr>
</tbody>
</table>
decisions, is essential for having working Shared Leadership. We decided to name this theme as Shared Leadership to adhere to agile teams’ characteristic of being self-organized.

V. DISCUSSION

This section discusses this study’s main findings and implications. Section IV presented this study’s resulting thematic network, which helps organize the knowledge regarding ASD Teamwork factors. It eases knowledge sharing by defining the terminology to be used for the ASD Teamwork factors. For instance, we identified three terms for the factor Redundancy: Backup, Backup Behavior, and Redundancy.

Further, it provides a better understanding of the interrelationships between the factors (i.e., objects) of ASD Teamwork’s knowledge field. However, notice that the proposed thematic network is not a cause-consequence model; instead, it comprehensively defines the themes (i.e., dimensions) that should be approached when describing ASD Teamwork. Moreover, the themes presented in Figure 1 respond our RQ defined in Section III.

It is intended to be used as a blueprint for both researchers and practitioners. However, it is not our intention to make the themes and their relations with ASD Teamwork something that must be followed as is, but to guide whoever intends to work towards their relevance based on literature findings.

The proposed thematic network also assists on identifying gaps in the ASD Teamwork knowledge field. Table I presents the distribution of occurrences for each of the identified ASD Teamwork factors. Notice that some factors have a high frequency, such as Performance Monitoring and Communication with nine appearances, while others only have one, such as Task Control and Awareness. Such results might indicate that the higher frequency factors are more important than ones with fewer, but it might only mean that they have been studied more. Thus, lines of research might follow from understanding the frequency distribution of the factors as shown in Table I.

Further, the proposed thematic network could be used to create a taxonomy for the ASD Teamwork field.

For researchers that intend to advance on the definition of ASD Teamwork constructs or its measurement, we expect them to consider all themes presented in our thematic network and complement or adapt their models/instruments based on the specific use-cases in which they fall. Taking previous studies into consideration when advancing the state-of-art is a premise. However, even the definition of another thematic network with the same purpose as the one defined in this study could bring valuable discussions around the topic that would benefit the academic community.

For practitioners, the thematic network can support their decision-making process. Practitioners can use it as a reference for understanding the factors and dimensions that comprise ASD Teamwork. With this, they can, for example, define mechanisms to monitor such dimensions and use the collected data as a reference to drive actions towards improving the team’s performance. Furthermore, they can also extend our thematic network to achieve tailored instances to the team’s context.

VI. THREATS TO VALIDITY

This section discusses this study’s threats to validity following the classification proposed by Wohlin et al. [21]: construct, internal, conclusion, and external validity.

- **Construct validity**: we analyzed the studies following a thematic analysis approach, in which multiple researchers participated to avoid researcher bias. However, we emphasize that removing researcher bias in qualitative research is virtually impossible. Thus, it is possible that the resulting thematic network (Figure 1) and codes (Table I) are not representative due to the researchers’ interpretations.
• **Internal validity**: to assure credibility in our findings, multiple researchers checked the extracted coding, themes, and the data presented in Figure 1.

• **Conclusion validity**: there is the risk that, since there is a threat to the construct validity of the thematic network, it influenced the extracted data and, consequently, our conclusions regarding the relationship between concepts.

• **External validity**: even though the data source was not a result of a broad literature review, we believe that it was representative of the state-of-the-art. Thus, we believe that the proposed thematic network is representative of agile TWQ in general. However, it might be possible that for specific situations, such as initial stages of agility adoption, including the transition from a traditional project management context, different factors could be included for ASD Teamwork.

**VII. CONCLUSION**

This paper presented a thematic network that relates the main dimensions of ASD Teamwork. The thematic network consists of four higher-level themes: Cohesion, Orientation, Shared Leadership, and Autonomy. It also consists of additional nine themes.

It contributes to the field by defining a common terminology to be used for ASD Teamwork. It also provides a better understanding of the interrelationships between the factors of ASD Teamwork knowledge field. Thus, it advances the state-of-art in regards to ASD Teamwork construct definition. Further, it can support practitioners’ decision-making process by being a reference for understanding the factors and dimensions of ASD Teamwork.

The main limitations of this study are related to the researchers’ bias regarding the execution of the thematic network process explained in Section III-B; and the inclusion and exclusion criteria restrictiveness of the forward snowballing to build the data source for the thematic network construction. However, we still believe that the missing factors would still fall into the themes that we have defined.

The study identifies several opportunities for future work, including using the proposed thematic network to build a Teamwork taxonomy and as the basis for potentially refining the Teamwork model constructs (or instruments) presented in Freire et al. [14], Moe et al. [10], and Lindsjørn et al. [16].

**REFERENCES**


A Comparative Study of Psychometric Instruments in Software Engineering

G. Guimarães¹, M.Perkusich¹, D. Albuquerque¹, E. Guimarães², H. Almeida¹, D. Santos¹ A. Perkusich¹
¹Federal University of Campina Grande - Intelligent Software Engineering Group (ISE/Virtus) - Paraiba, Brazil
²The Pennsylvania State University - Malvern, Pennsylvania - USA

Abstract—Over the years, researchers have explored the influence of human factors in software engineering, showing that the team members’ personalities might affect teamwork. However, it is challenging to measure software engineers’ personalities due to the number of available psychometric instruments and the possibility of using different scales and classifications. Our study compares the personality traits measured by three psychometric instruments used in Software Engineering: Big Five Inventory (BFI), 16 Personality Factors (16PF), and Context Cards (CC). For this purpose, we executed an empirical study in which we collected data from 29 software developers for each of the evaluated instruments. As a result, we identified a moderate correlation between BFI and 16PF, confirming the state-of-the-art. For the remaining combinations, there was a weak correlation. As implications for this research, there is a need to empirically evaluate BFI and CC (context-specific survey) in terms of construct validity since they have moderate to low correlation.

Index Terms—Human Aspects, Social Aspects, Personality, Software Engineering, Psychometric Instruments.

I. INTRODUCTION

The success of software development projects is directly related to the team members’ technical (a.k.a. Hard skills) and non-technical skills (a.k.a Soft Skills) [1]. Soft skills are becoming more important in the industrial environment because they affect team cohesion and team climate [2], [3], impacting its productivity and outcomes’ quality [4], [5].

A key aspect of studying soft skills is personality. Many studies investigated the effects of personality on teamwork performance in the last forty years [6], [7]. These studies used several psychometric instruments to evaluate personality types and personality traits of software engineers [8], [6], [7]. The studies mostly used the Myers-Briggs Type Indicator (MBTI) or tests based on the Big Five (BF), such as Big Five Inventory (BFI) and Revised NEO Personality Inventory (NEO-PI-R). However, other psychometric instruments were also used, such as International Personality Item Pool (IPIP), the Sixteen Personality Factor Questionnaire (16PF), and Context-Specific Survey Instrument (Context Cards) [6].

Choosing a psychometric instrument is not straightforward because some require training and a license to be used. Moreover, McDonald and Edwards reported misuse of personality tests in software engineering (SE) [9]. They argued that the inappropriate use of psychological tests and fundamental misunderstandings of personality theory caused a lack of progress in this field. Additionally, Graziotin et al. [10] demonstrated a deeper confusion on assessing related constructs using personality tests. For example, Capretz and Ahmed [11] considered that the introvert trait is suitable for the programmer role, whereas Gorla and Lam [12] concluded that it is the Extrovert trait.

Having reliable data is the most critical factor for any SE measurement approach. Such observation is also valid for psychometric instruments. The inadequate application of psychometric instruments and their interpretation might lead to invalid results, economic loss, and harm to individuals. If the psychometric instruments or their usage are not valid, all the resulting conclusions are also invalid. Psychometric instruments measure latent variables (i.e., unobservable constructs) such as intelligence, personality, and happiness. Therefore, evaluating the psychometric instrument used is crucial to ensure that the variables are measured correctly.

Cruz et al. [6] discusses the existence of many disagreements in the SE research community regarding (i) the application of psychometric instruments and (ii) the interpretation of their results, comparing psychometric instruments’ constructs to understand how to measure software engineers’ personalities and their impact on productivity and quality. However, no other works are performing a similar comparison analysis in the past five years.

Moreover, Gulati et al. [13] examined studies based on human factors in software engineering. They compared studies relating to different personality instruments (i.e., MBTI, KTS, and BFI). Balijepally et al. [14] focused their research on comparing two emerging models, BFI and MBTI, for assessing personality traits in SE. To the best of our knowledge, these studies promote a discussion relating to software engineering and psychology but do not explore the correlation analysis among the personality instruments.

To address this gap, we investigated the similarity between three psychometric instruments: Big Five Inventory (BFI), 16 Personality Factors (16PF), and Context Cards (CC). We used these instruments because (i) studies in SE recurrently use them, (ii) they are of the public domain or readily available for researchers, (iii) they are clear to use in SE, and their data analysis have a vocabulary that is easy to understand, (iv) they do not have a large number of items, thus easing its execution and interpretation. We collected data from 29 subjects with about three to five years of experience in the area, and most of them were developers of web or mobile projects. We compared the instruments in terms of the number of questions, option type, response time, and the Kendall correlation.
This paper details the applied study and summarizes the results regarding the similarity of the evaluated psychometric instruments. The remainder of this paper is organized as follows. Section II provides a background with an overview of psychometric instruments for SE. Section III describes the study design. Section IV presents the results and discusses the answers to the research questions. Section V analyzes the study’s threats to validity. Finally, Section VI presents our conclusions and directions to future work.

II. BACKGROUND AND RELATED WORK

This section presents an overview of psychometric instruments and describes the studies that compared psychometric instruments in the context of Software Engineering.

Psychometric Instruments Overview. Personality involves different theoretical perspectives, definitions, and levels of abstraction. We used the personality definition established by Ryckman [15], in which it is defined as “a dynamic and organized set of characteristics possessed by a person that uniquely influences his or her cognitions, motivations, and behaviors in various situations”. We used this definition due to its popularity in Software Engineering research.

Psychometric instruments have been used to evaluate personality traits and types of individuals, usually by using questionnaires. Personality traits are stable characteristics of individuals such as being optimistic, sociable, and imaginative, whereas Personality types are constructs that indicate independent groups such as mediator, entrepreneur, and adventurer. For instance, using the 16PF psychometric instrument, a person that scores high on the traits “introversion”, “sensing”, “thinking”, and “judging” would be classified as being of the personality type “logistician”.

Psychometric Instruments in SE. Software development organizations use psychometric instruments to measure their members’ personality [16], [17]. The most used psychometric instruments in SE are based on the Big Five (BF) theory - e.g., Big Five Inventory (BFI) [6]. Other psychometric instruments have been widely used in SE research, such as the 16 Personality Factor Questionnaire (16PF) and Context-specific survey instrument (CC). Next, we describe each of the aforementioned psychometric instruments.

Big Five Inventory (BFI) describes the personality by employing broad factors (dimensions) of personality traits [18]. Its five dimensions are Extraversion, Agreeableness, Conscientiousness, Neuroticism (a.k.a. Emotional Stability), and Openness to Experience (sometimes called Intellect or Imagination). The BFI-44 is a self-report inventory created to measure the Big Five dimensions. The psychometric instrument contains 44 items and consists of short and descriptive phrases that respondents rated on a 5-point scale ranging from strong disagreement to strong agreement. This method is not in the public domain. However, it is readily available for researchers to use for non-commercial research purposes.

The 16 Personality Factor Questionnaire (16PF) is a psychometric instrument to identify characteristics, personality traits, and behavior. 16PF was published in 1949 and has been used to evaluate personality in many contexts, including career assessment and SE [4]. The 16PF online test comprises five personality dimensions, making up 16 personality types. 16PF has five aspects: Mind, Energy, Nature, Tactic, and Identity. 16PF generates 16 types of personalities by acronyms generated from the dichotomies emitted by the psychometric instrument’s aspects. The combination of four personality aspects results in a personality type. For instance, the combination of Extraversion (E), Observant (S), Thinking (T), Judging (J), and Assertive (-A) result in the personality ESTJ-A. The Identity scale (i.e., assertive or turbulent) is in all personality types because it affects other scales. As a result, when we considered this scale, the method describes 32 different personality types.

Context-specific survey instrument (CC) was proposed by Yilmaz et al. [8] aims to reveal and illustrate the personality characteristics of the individuals in software development teams. The instrument combines situations from companies with basic patterns (items) of the Big Five Inventories (BFI-44) questionnaire to create a card game-based personality identification method [8]. The context cards describe the human personality traits in terms of five fundamental factors: Extraversion, Openness, Agreeableness, Neuroticism, and Conscientiousness. This model identified six themes (traits) for each factor, totaling 30 themes. For example, the factor Extroversion has the traits talkative, assertive, energetic, active, approachable, and outgoing.

Psychometric Instruments Evaluation in SE. Next, we discuss studies that assessed psychometric instruments in SE. Jia et al. [19] reviewed and compared three psychometric instruments (i.e., BFI, MBTI, and KTS). They observed the number of questions, option type, and time spent answering the test. The researchers collected empirical evidence for comparison from articles published between 2010 and 2014. They concluded that BFI is the more suitable alternative to evaluate soft-skills in software development activities.

Another study by Gulati et al. [13] compared studies published between 2003 and 2014 that analyzed human factors in software engineering. They concluded that the most popular psychometric instruments in SE are MBTI and BFI. Finally, Balijepally et al. [14] dedicated their research mainly to compare BFI and MBTI. They suggested that BFI is more valuable than MBTI because BFI provides better measures for all MBTI factors, and it also evaluates Neuroticism, an important personality trait.

Even though these studies promote a great discussion relating to software development activities (i.e., SE area) and soft-skills (i.e., Psychology area), they perform the models’ comparison based solely on data available in the literature. Consequently, we conclude that these studies highlight the importance of human psychology in SE and realized conceptual comparisons but lack evidence about how the psychometric instruments compare in the context of SE. To address this gap, we compared personality instruments by collecting data from software developers and analyzing the correlation between the instruments’ answers.
III. Research Methodology

This section presents the research methodology for the empirical study, including research questions, subjects’ profiles, experimental materials, and data analysis procedures.

Objective and Research Questions. Our study aimed to analyze and evaluate the correlation between three state-of-the-art psychometric instruments (i.e., BFI, 16PF, and CC) in the context of software development activities. Given this, we formulated some Research Questions (RQs):

- **RQ1:** To what extent do BFI dimensions correlate with 16PF considering software developers’ personality?
- **RQ2:** To what extent do 16PF aspects correlate with CC factors considering software developers’ personality?
- **RQ3:** To what extent do BFI dimensions correlate with CC factors considering software developers’ personality?

Subjects. We applied the psychometric instruments with 29 software developers (24 men and five women) from one Brazilian software organization. This organization had approximately 250 employees and produced more than 40 projects in collaboration with multinational partners. The employees were organized into small agile teams (around five to ten members).

The participant’s ages ranged from 21 to 29 years, with a mean of 24 years. They had an average of three years of experience with software development. They developed Web and mobile applications using different technologies (e.g., Javascript, Java, HTML). Overall, the subjects’ profile meets our study assumptions since all of them work in software development. They participated voluntarily in the study and had no experience with the instruments used.

Experimental Materials. We created a questionnaire with five sections to apply the psychometric instruments and submitted it to the ethics committee from the Federal University of Campina Grande (UFCG) for analysis and approval before conducting the study. The ethics committee gave us the certification (02505718.0.0000.5182), meaning we could collect data using the questionnaire. In the following, we present information about the questionnaire.

The **first section** of the questionnaire presented the study objectives and the consent form, as approved by the ethics committee. This section contained information to motivate the participants to participate in the study. The **second section** contained questions to collect demographic data, including name, gender, experiences, and age. The **third section** contained BFI-44 questions translated and adapted to Portuguese by Andrade [20]. We also consulted the BFI-44 in Portuguese available on the Berkeley Personality Lab site [21]. All questions were answered through a five-point Likert-type scale: 1 (Strongly disagree) - 5 (Strongly agree), expressing their agreement degree regarding the question’s descriptions. The **fourth section** contained 16PF questions. The 16PF has about 60 questions (statements), each of them to be answered through a seven-point Likert-type scale (from “agree” to “disagree”).

Finally, the **fifth section** included the CC questions. We translated and adapted the Context Cards to Portuguese. This psychometric instrument includes 60 cards (e.g., situations, questions), in which each card presents a situation and two optional answers described by A and B. The optional answers describe SE situations. One example of the original text on the one card is presented next. Situation (“During a team base discussion...”), Option A (“Evidence suggests that what we learn is mostly from our conflicts.”), and Option B (“I believe compromise between people for a common ground more successful.”). Each person spends about 40 minutes - on average - to complete the questionnaire.

Procedure. We applied the questionnaire during the period that the software organization made available for the research. Before the questionnaire application, the study’s authors ran a training session with the participants. The training session lasted 30 minutes, and the participants spent between 30 to 40 minutes answering the entire questionnaire. The training session aimed to present the questionnaires’ concepts and level the participants’ understanding regarding the psychometric instruments and study’s goals. In other words, we motivated the participants to answer based on reality (not intentions) and that there were no right or wrong answers. Further, we emphasized that they would not be identified. Such support minimized internal validity threats.

Analysis Procedure. We defined the following criteria for comparing the psychometric instruments: the number of questions, option type, response time, and the correlation between the results (i.e., the correlation between the facets, dimensions, and aspects of psychometric instruments). The correlation analysis sought to verify the correlation between the personality traits presented by instruments for each participant. The BFI-44, 16PF, and Context Cards instruments contain different sets of Likert-type items combined into single composite scores (as explained in Section II). Thus, they calculate a score for each facet or dimension of the psychometric instrument. Each of these composite scores provides a quantitative measure of a personality trait.

Considering that the variables measured in this study are ordinal, we used the Kendall correlation coefficient. Kendall correlation coefficient is a nonparametric measure of the strength and direction of the association between two variables measured on at least an ordinal scale [21]. We also used the correlation coefficients interpretation for psychology described by Dancey and Reidy [22].

IV. Results and Discussion

This section presents the main results for the comparison between the three psychometric instruments.

Personality Test Results. We obtained personality traits’ scores for each participant by applying the three psychometric instruments (i.e., BFI, 16PF, and CC). It is worth mentioning that each instrument provided a set of personality trait scores for each participant. We calculated these scores using the guidelines for each instrument. Next, we present the results when applying each psychometric instrument.
**BFI results**: We found the following quantities of participants with the dimension value assessed above 50%: Extraversion (20 participants), Agreeableness (29 participants), Conscientiousness (28 participants), Neuroticism (12 participants), and Openness to Experience (24 participants).

**16PF results**: We found more Extroverts (52%) than Introverts (48%), more Sensing (55%) than Intuitive (45%), fairly more Feeling (65%) than Thinking (35%), and more Judging (65%) compared to Perceiving (35%) type. The personality types most present were ENFJ, ESFJ, and ISFJ. However, we did not have participants with the personality types ISTP, ESTP, INJF, and ENTP. Considering the Identity scale, the personality type most present was ENFI-A.

**CC results**: We obtained the following quantities of participants with the factor value assessed above 50%: Extraversion (17 participants), Agreeableness (27 participants), Conscientiousness (15 participants), Neuroticism (4 participants), and Openness to Experience (8 participants).

We analyzed the number of questions, option type, and time to respond to the psychometric instruments compared. Considering the number of questions and option type, 16PF contains 60 7-point questions, BFI-44 contains 44 5-point questions, and CC contains 44 A or B questions. The participant’s average response time was 12 minutes to 16PF, 10 minutes to BFI-44, and 15 minutes to CC.

Nevertheless, in addition to these comparison criteria, we were interested in the correlation between the test results (i.e., the correlation between the facets, dimensions, and aspects of psychometric instruments). We discussed the analysis of this criterion by answering the research questions.

**RQ1 - To what extent do BFI dimensions correlate with 16PF aspects considering software developers’ personality?**. Aims to address this RQ, we applied Kendall’s correlation between 16PF dichotomies and BFI dimensions. The correlation results can be seen in Table I.

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>CORRELATION BETWEEN 16PF DICHOTOMIES AND BFI DIMENSIONS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>16PF</td>
<td>Extrovert</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feeling</td>
</tr>
<tr>
<td></td>
<td>Intuitive</td>
</tr>
<tr>
<td>16PF</td>
<td>Judging</td>
</tr>
<tr>
<td></td>
<td>Assertive</td>
</tr>
</tbody>
</table>

Figure 1 shows the distribution of participants considering the 16PF’s Extroversion dichotomy (16PF-E), 16PF’s Introversion dichotomy (16PF-I), and the BFI’s Extraversion dimension (BFI-E). In the figure, each point represents the participant score (i.e., the score of dimension or facet) obtained from the psychometric instrument.

We found a moderate correlation between BFI’s Extraversion dimension (BFI-E) and the 16PF’s Mind aspect (16PF-E and 16PF-I). This correlation was positive to 16PF’s Extroversion dichotomy (16PF-E) and negative to 16PF’s Introversion dichotomy (16PF-I). The coefficient of 0.571 (p < 0.01) indicates a moderate positive correlation between 16PF-E and BFI-E. As expected, the correlation for 16PF-I was inversely correlated -0.571 (p < 0.01) with BFI-E. The results found reinforced the analysis performed by Jia et al. [19], [14], which stated that BFI’s Extraversion dimension (BFI-E) is correlated to the 16PF’s Extraversion dichotomy (16PF-E).

Considering the distribution of participants for 16PF’s Feeling dichotomy (16PF-F) and 16PF’s Thinking dichotomy (16PF-T) and the BFI’s Agreeableness dimension (BFI-A), We found a moderate correlation with a coefficient of 0.534 (p < 0.01) between 16PF-F and BFI-A. As expected, the correlation between 16PF-T and BFI-A was inversely correlated. Also, we found a moderate correlation between BFI’s Conscientiousness dimension (BFI-C) and the 16PF’s Identity aspect. This correlation was positive to 16PF’s Assertive dichotomy (16PF-A) and negative to 16PF’s Turbulent dichotomy (16PF-T).

Similarly, we obtained a moderate correlation between BFI’s “Openness to Experience” dimension (BFI-O) and the 16PF’s Energy aspect (16PF-N and 16PF-S). This correlation was positive to 16PF’s Intuitive dichotomy (16PF-N) and negative to 16PF’s Observant dichotomy (16PF-S). The correlation coefficient of 0.502 (p < 0.01) indicated a moderate correlation between the BFI-O and the 16PF’s Intuitive dichotomy (16PF-N). As expected, the correlation coefficient was negative between the BFI’s “Openness to Experience” dimension (BFI-O) and the 16PF’s Observant dichotomy (16PF-S).

Considering the 16PF’s Judging dichotomy (16PF-J), 16PF’s Prospecting dichotomy (16PF-P), and the BFI’s Conscientiousness dimension (BFI-C). The correlation coefficient of 0.712 (p < 0.01) indicated a strong positive correlation between 16PF-J and BFI-C. As expected, the correlation between 16PF-P and BFI-A was inversely correlated -0.712 (p < 0.01). We obtained a strong correlation between BFI-C and the 16PF’s Tactic aspect (16PF-J and 16PF-P). This correlation was positive with 16PF-J and negative to 16PF-P.

As in previous studies [23], [24], [25], [26], our study identified a strong correlation between 16PF-J and BFI-C. However, we found a moderate correlation between 16PF-E and BFI-E, 16PF-F and BFI-A, and 16PF-N and BFI-O, unlike Cattell and Mead [23], which found a strong correlation. Additionally, we observed a weak correlation between 16PF-F...
and BFI-N.

Further, we analyzed the correlation of 16PF’s Identify aspect (16PF-A and 16PF-T) with BFI’s dimensions. We obtained a coefficient of -0.578 (p < 0.01) for the correlation between the 16PF’s Assertive dichotomy (16PF-A) and BFI’s Neuroticism dimension (BFI-N). Further, the correlation between (16PF-A) and BFI’s Conscientiousness dimension (BFI-C) was 0.40 (p < 0.01). Considering the 16PF’s Turbulent dichotomy (16PF-T), we obtained a correlation coefficient of 0.580 (p < 0.01) with BFI-N, and -0.416 (p < 0.01) with BFI-C. Table I illustrates the results of Kendall’s correlation between 16PF dichotomies and BFI dimensions.

We concluded that BFI-C correlated strongly with 16PF-E. We found a moderate correlation between 16PF-E and BFI-E, 16PF-F and BFI-A, and 16PF-N and BFI-O. We found a weak correlation between 16PF-F and BFI-N. Further, 16PF-A correlated moderately with BFI-C (moderately positive) and BFI-N (moderately negative). Thus, we concluded that BFI and 16PF had a moderate correlation.

RQ2 - To what extent do 16PF aspects correlate with CC factors considering software developers’ personality? Aims to address this RQ, we applied Kendall’s correlation between 16PF dichotomies and CC factors. The correlation results can be seen in Table II. The correlation coefficient of 0.35 (p < 0.01) indicates a weak positive correlation between 16PF’s Extroversion dichotomy (16PF-E) and the CC’s Extroversion factor (CC-E). Additionally, we obtained a weak correlation between CC-E and 16PF’s Mind aspect (16PF-E and 16PF-I). Similar to what happens with 16PF and BFI.

Regarding 16PF’s Judging dichotomy (16PF-J) and CC’s “Openness to Experience” factor (CC-O), we found a negative correlation -0.36 (p < 0.01) between 16PF-J and CC-O. Further, we found a weak correlation between CC’s “Openness to Experience” factor (CC-O) and the 16PF’s Tactic aspect (16PF-J and 16PF-P). Considering the 16PF’s Feeling dichotomy (16PF-F) and CC’s Neuroticism factor (CC-N), we observed a negative correlation coefficient of -0.34 (p < 0.01), which indicated the weak correlation between these characteristics. Finally, we found a weak correlation between CC’s Neuroticism factor (CC-N) and 16PF’s Nature aspect (16PF-T and 16PF-F). Given the discussed results, we concluded that the correlation between 16PF-J and CC was weak.

RQ3 - To what extent do BFI dimensions correlate with CC factors considering software developers’ personality? Aiming to address this RQ, we applied Kendall’s correlation between BFI dimensions and CC factors. The correlation results can be seen in Table III. Considering the five characteristics analyzed for each psychometric instrument, we had a weak correlation (correlation coefficient of 0.38 (p < 0.01)) between CC’s Extroversion factor (CC-E) and BFI’s Extraversion dimension (BFI-E). We found a correlation coefficient of 0.30 (p < 0.01) between BFI’s Conscientiousness dimension (BFI-C) and CC’s Extroversion factor (CC-E). Further, we obtained a weak correlation between CC’s Extroversion factor (CC-E) and the BFI’s Extraversion dimension (BFI-E). CC’s Conscientiousness factor (CC-C) had a correlation coefficient of -0.34 (p < 0.01) with BFI’s Agreeableness dimension (BFI-A) and 0.32 (p < 0.01) with BFI’s Neuroticism dimension (BFI-N). Thus, we concluded that there was a weak correlation between CC’s Conscientiousness factor (CC-C) and the BFI’s Agreeableness dimension (BFI-A) and BFI’s Neuroticism dimension (BFI-N).

TABLE III

| BFI Dimensions | CC Factors | Context Cards | Table III
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.27</td>
<td>0.24</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Despite measuring similar characteristics, we concluded that the relationship between BFI and CC is weak. These results are not similar to those found by Yilmaz et al. [8], in which the authors proposed and validated CC. In this study, they proposed results that assumed a strong correlation between the use of CC and BFI. Such correlations were the CC’s Extroversion factor (CC-E) with the BFI’s Extraversion dimension (BFI-E) (correlation coefficient of 0.93). They also found a strong correlation between CC’s Conscientiousness factor (CC-E) and BFI’s Conscientiousness dimension (correlation coefficient of 0.79).

We believe that a possible explanation for such distinction between the study findings and Yilmaz et al.’s [8] maybe the loss of semantics aspects from the original CC (once the originals were written in English language and the ones applied here were translated into Brazilian Portuguese). Besides that, the cultural factor may have influenced the interviewees’ understanding of some questionnaire items’ situations. We concluded that CC inventory is currently lacking independent replications to validate its reliability. In sum, this highlights the importance of developing such research in the SE field to enhance the surveys and studies that approach psychometric aspects and team formation.

V. Threats to Validity

This section discusses the study’s threats to validity following the classification proposed by Wohlin et al. [27] and the strategies applied to mitigate them.

Internal validity: We applied the questionnaire during the period that the company made available for the research. This session lasted around 50 minutes, which may have
influenced the results due to fatigue. Another threat is related to understanding each of the psychometric instruments used in the study. The first and second authors ran a training session with the study’s subjects to mitigate this threat. Conclusion validity: We obtained the results from the data using the Kendall correlation coefficient. We also adopted a free software for statistical computing and interpreted the correlation coefficients using psychology guidelines proposed in [22].

Construct validity: We used psychometric instruments previously validated by other studies and mapped 16PF aspects, BFI types, and personality factors CC based on the literature. However, the translation into Portuguese has removed essential aspects of measurement for the CC. The SE scenarios can be regionalized, not matching the scenarios of the participants in this study. External validity: the relatively small sample size could limit external validity. Therefore, this study should be replicated with a larger sample size to confirm the initial results and address external validity issues. Lastly, the generalizability of these results is subject to certain limitations.

VI. CONCLUSIONS AND FUTURE WORKS

This paper presented the results of an empirical comparison of three psychometric instruments (i.e., 16PF, BFI, and CC) in SE. Our results showed a moderate correlation between 16PF and BFI, a weak correlation between (i) 16PF and CC, and (ii) BFI and CC. In terms of construct analysis (i.e., psychological perspective), our study reinforces the results of the Psychology field with regards to the correlation between 16PF and BFI, and we found a weak correlation between CC and BFI, opposing the findings from Yilmaz et al. [8]. Since this is an emerging field in SE, such contradicting results are expected in the scientific process. The analysis was performed in light of the instruments’ psychological constructs and within the SE field, having relevance for both fields.

Regarding the implications for practice, having a better understanding of psychometric instruments might help hire and train software engineers and form software teams, but this analysis requires further studies. We expect that the present research contributes to having fewer contradictions between the psychometric test results in SE in the future. As future work, we plan to expand our study by comparing other psychometric instruments using different criteria or observing the application of psychometric instruments with more participants.

REFERENCES

Evaluating a Bayesian Network to Predict Customer Satisfaction in Scrum Software Development Projects: An Empirical Study with One Company

Mirko Perkusich
Intelligent Software Engineering Group
mirko@virtus.ufcg.edu.br

Gleyser Guimaraes
Federal University of Campina Grande
Intelligent Software Engineering Group
gleyser@copin.ufcg.edu.br

Kyller Gorgonio
Federal University of Campina Grande
Intelligent Software Engineering Group
kyller@copin.ufcg.edu.br

Hyggo Almeida
Federal University of Campina Grande
Intelligent Software Engineering Group
hyggo@dsc.ufcg.edu.br

Angelo Perkusich
Federal University of Campina Grande
Intelligent Software Engineering Group
perkusic@dee.ufcg.edu.br

Abstract—Using knowledge-based systems for helping agile teams to improve their performance is not a fact in the industry. In previous work, we have presented Kaizen, a knowledge-based Bayesian network for assisting Scrum teams in diagnosing their value stream in light of the predicted Customer Satisfaction and, consequently, improve their performance. This study assesses Kaizen’s accuracy to predict Customer Satisfaction using real-world data. We adopted Kaizen for one software development company and collected data from 18 projects using an online questionnaire. We collected two types of data: inputs for Kaizen and the expected Customer satisfaction. We used the first type of collected data as inputs for Kaizen to calculate the predicted Customer satisfaction. Then, we assessed Kaizen’s accuracy by comparing the predicted (i.e., calculated) and expected (i.e., collected) Customer satisfaction using face value and the average Brier score. Considering the face value, Kaizen predicted Customer Satisfaction correctly for 14 out of the 18 projects. The average Brier Score was 0.16. The model predicts, with satisfactory accuracy, the Customer Satisfaction and systemizes the process for Scrum teams to self-diagnose, enabling for causal analysis and supporting their continuous improvement.

Index Terms—Agile Software Development; Questionnaire Survey; Bayesian Network.

I. INTRODUCTION

Using knowledge-based systems for helping agile teams to improve their performance is not a fact in the industry. In previous work, we have presented a knowledge-based Bayesian network to assist Scrum teams in diagnosing their value stream [14], [13], [12], and, consequently, improve their performance. From here on, we refer to such a Bayesian network as Kaizen. Kaizen has as the “target variable” the Customer Satisfaction. Thus, Kaizen assists Scrum teams to improve their efficiency in light of the predicted customer satisfaction. In previous studies, we have validated it with simulated scenarios (i.e., model walkthrough) [13] and by analyzing its ability to identify 14 Scrum anti-patterns [12] described in the literature [4].

Further, we have evaluated its practical utility by using it in two projects from one software company [14]. However, such a case study focused on Kaizen’s ability to support decision-making, not on its predictive accuracy. Thus, at this point, its accuracy for predicting Customer Satisfaction has not been assessed with real-world data.

This study addresses this gap by adopting Kaizen for one software development company. The adoption process included complementing it with a questionnaire to collect projects’ data from their Scrum Masters and customer representatives. As a result, we collected data from 18 projects. We collected two data types for each project: inputs for Kaizen and the expected Customer satisfaction. We used the first type (inputs for Kaizen) to calculate the predicted Customer satisfaction. Then, we assessed Kaizen’s accuracy by comparing the predicted and expected Customer satisfaction using face value and the average Brier score. This paper reports the employed methodology and our results. Further, it shows an example of how Scrum teams can adopt Kaizen.

This paper is organized as follows. Section II presents the necessary background on the Bayesian network and our reasoning for selecting it as the modeling tool for Kaizen. Section III presents an overview of Kaizen. Section IV presents the employed research methodology. Section V discusses our results, their implications, and this study’s threats to validity. Finally, Section VI presents our final remarks and directions to future work.

II. BAYESIAN NETWORKS

Bayesian networks are probabilistic graph models that represent knowledge about an uncertain domain. A Bayesian network, $B$, is a directed acyclic graph representing a joint probability distribution over a set of random variables $V$. The network is defined by the pair $B = \{G, \Theta\}$. $G$ is the directed acyclic graph in which the nodes $X_1, \ldots, X_n$ represent random variables, and the arcs represent the direct dependencies between these variables. Therefore, a Bayesian network consists of two parts: the directed acyclic graph and the probability distributions. They can be constructed based on domain experts’ knowledge, statistical techniques, or both (i.e., hybrid). Since the data needed to construct Kaizen was not available publicly, we relied on domain experts’ knowledge for both parts of our model [14], [13], [12].
Using Bayesian networks for supporting decision-making in software engineering has the following advantages.

- In a single model, it is possible to maintain observations, statistical distributions, prior assumptions, and expert judgment.
- Allows the encoding of causal relationships among variables for prognosis.
- Enables human diagnosis due to its explanatory nature (in contrast with opaque neural networks that are hard to analyze and test), reducing its risk of adoption.
- Handles missing data.

III. KAIZEN OVERVIEW

This section presents an overview of Kaizen. Its purpose is to present a high-level view of its structure. Notice that the methods applied to construct it and its details are presented elsewhere [14], [13], [12].

The reasoning behind the model is based on the Scrum Guide’s vision that Scrum is a means for transforming ideas into value [18]. Further, according to the Evidence-Based Management Guide, there are four dimensions four Key Value Areas: Current Value, Time to Market, Unrealized Value, and Ability to Innovate. Kaizen focuses on Current Value; more specifically, it assesses Scrum’s adoption in light of the value perceived by customers or the product’s users. Its main requirement was to model Scrum’s product delivery mechanisms, from conceptualization to release, that affect the probability of delivering products that satisfy customers.

Given this, Kaizen’s “target variable” is Customer satisfaction. Kaizen’s goal is to be used by Scrum teams to improve their product delivery process, having the predicted Customer satisfaction levels as the reference. It enables Scrum teams to predict Customer satisfaction given their current practices and diagnose themselves, enabling early detection of undesired deviations. Thus, Kaizen’s variables model the team’s value stream, in other words, its procedures to deliver value to customers.

The process of building Kaizen was top-down, decomposing Customer satisfaction into attributes that the team could observe. Since, in Scrum, the ideas are stored in the Product Backlog, and the value is delivered through the release of Increment, these are the variables to “predict” Customer satisfaction.

Figure 1 presents the Kaizen’s graph. Figure 1, a white node represents a node not directly observable; an orange node represents a predictor; and a green one, an indicator. A predictor is a factor that can be observed to predict the value of its child node. An indicator is a factor that can be observed to measure the current value of its parent node. The difference between a predictor and an indicator is the time of measurement. For instance, consider Sprint Planning. Going into such an event, knowing the Projected Capacity and Past Performance increases the chances of having good Sprint Planning, and its success can be measured by the quality of the resulting artifact: the Sprint Backlog.

Notice that Kaizen does not include two of Scrum’s key aspects: Sprint Retrospective and Scrum Master. The reason is that Kaizen was built considering the Scrum Master’s perspective and that it should be used during Sprint Retrospective events. Further, it is worthy of mentioning that Kaizen we built by extracting knowledge available on the literature and tacit knowledge from industry experts following the Expert-Based Knowledge Engineering of Bayesian Networks methodology [9]. A file for executing Kaizen in AgenaRisk\(^1\), the Bayesian network inference system in which Kaizen’s model executes, is made available online\(^2\).

IV. RESEARCH METHODOLOGY

This study’s goal was to analyze if Kaizen measures what it is supposed to measure (i.e., customer satisfaction level). Further, it is essential to mention that our study’s results are limited to the company’s context from which we collected data. Given this, we explored the following research question.

RQ What is Kaizen’s accuracy for predicting the customer satisfaction level’s for the company under study?

As measures for predictive accuracy, we used face value (i.e., outcome adequacy [10]) and the Brier score. Face value measures if the customer satisfaction level calculated by Kaizen provides the highest probability to a state that matches the value expected, given the data collected from the company. For instance, if the expected value for customer satisfaction is Low, and the predicted value for customer satisfaction is \( \text{Low} = 0.25 \), \( \text{Moderate} = 0.60 \), and \( \text{High} = 0.15 \), FV is \( \text{FALSE} \) (or 0).

However, using the face value is not enough because one would agree that predicting the correct value with 91% is better than, for instance, 50%. Thus, we also used the Brier score, which is the mean squared difference between the predicted probability and observed outcome. A Brier score of 0 indicates a perfect model, and 1 is the worst score possible. We calculated the Brier score for each project and used the average to assess Kaizen’s predictive accuracy.

Next, Section IV-A contextualizes the company for which we collected data. Section IV-B describes the procedures employed to collect data.

A. Context Characterization

This section presents information to characterize the company under study following the context facets described by Petersen and Wohlin [15]: product, process, practices and techniques, people, organization and market.

The company under study is a research, development, and innovation center. It supplies services to industry partners in the context of several technological domains, including artificial intelligence, Web systems, and cyber-physical systems. It manages its initiatives through projects lasting around ten and eighteen months.

In general, the projects are executed using agile approaches such as Scrum or Kanban. The development practices and tools follow the guidelines defined by the organization but are adapted given the projects’ needs (e.g., programming language and type of system).

\(^1\)https://www.agenarisk.com/
\(^2\)https://doi.org/10.5281/zenodo.4604017
B. Data collection Procedure

Kaizen does not claim external validity [12] and adopting it requires the users to adapt it to their context for the same reasons that companies tailor methods (including Scrum) to fit their values, culture, reality, needs, and strategies [7], [1].

One of the steps necessary to adopt Kaizen is to map the variables of interest into data sources so that the users do not need to input data directly into the Bayesian network’s variables. Such data sources might be a questionnaire or a tool such as Sonarqube [17]. For instance, the variable Development and Testing Process Quality can be mapped to one or more questions on a questionnaire or use the technical debt indicators provided by tools such as Sonarqube.

For our study, our variables of interest are the “target variable”, Customer support and the leaf nodes (i.e., the ones marked with the color orange in Figure 1), which we mapped into two questionnaires. One questionnaire was aimed only to collect data for Customer support, and answered by the projects’ customer representatives. The projects’ Scrum Masters answered the other questionnaire to collect data for the remaining variables previously mentioned. For each variable of interest, we defined one question. The questionnaire made available for the Scrum Master also data about his/her profile and the project’s context. Section V-A details the process employed to recruit participants.

We designed and executed our study following the instructions presented by Runeson and Höst [16] and Molléri et al. [11]. Before sending the questionnaire to the customer representatives and Scrum Masters, we evaluated it through a pre-test with eleven participants, including people from the target population and colleagues with Scrum experience. The pre-test’s goal was to validate the questionnaire’s understandability. To execute the pre-test, each person responded to the questionnaires separately. Afterward, we executed a focus group to define the questionnaire’s final version.

As a result of the pre-test, for some questions, we created a glossary to support the respondents of the questionnaire and avoid misunderstandings regarding the used terminology. The questions followed a pattern. For some questions, we directly asked the user to assess each of the target variables’ current state using a 3-point Likert scale. We used such a scale because Kaizen’s variables use ordinal variables with three states. For instance, for the variable “Complexity”, the question was: “How complex are the requirements?”, with the alternatives “Low”, “Moderate”, and “High”. For others, we presented a sentence and asked their level of agreement (i.e., Disagree,
Neutral, Agree). There were also three cases in which we customized the possible answers to ease their understandability; an example is shown in Table I. The Supplementary Material, available online\(^3\), presents all questionnaires in their original language (i.e., Portuguese) and their translations into English.

The questionnaires were made available online using a Google Form\(^4\) and sent by the respondents through e-mail. We gave the respondents one week to answer the questionnaires, and the first author remained available if they had any doubts. All the participants that had agreed to participate in the study answered the questionnaires.

Once the answers were collected, we inputted the collected data into a Google Spreadsheet. Then, manually, we inputted the collected data into Kaizen’s model’s leaf nodes using AgenaRisk. AgenaRisk has a feature named “scenario”, which enables the user to tag a set of inputs for the model’s nodes (i.e., AgenaRisk calls each input into a node as an “evidence”). Thus, we created one “scenario” for each project, containing the set of “evidence” for each of the model’s leaf nodes, which followed directly from the questionnaire’s answers. For instance, the node Monitoring was mapped the question The Developers are monitoring tasks properly.. Thus, if for a given project, the answer to this question was Disagree, we inputted the “evidence” Low into the node Monitoring. We used the answers related to the variable Customer Satisfaction as the reference for calculating Kaizen’s predictive accuracy, discussed in Section V-B.

V. RESULTS AND DISCUSSION

This section describes the units of analysis (i.e., projects) and subjects (i.e., people who answered the questionnaires) (see Section V-A), discusses the results of Kaizen’s predictive accuracy (see Section V-B), presents its implications (see Section V-C), and the study’s threats to validity (see Section V-D).

A. Units of Analysis and Subjects

At the time of the data collection, the company had more than 30 projects being executed. Our goal was to collect data from as many projects as possible that used Scrum and focused on delivering software products or prototypes. The recruiting process started by having the first author directly inviting Scrum Masters and customer representatives from projects with the desired characteristics to participate in this study by explaining its goals and how it could benefit them. The first author explained that the data would be kept anonymous. Further, the first author explained that, after the data analysis was done, the original data, which mapped the project identification with the collected, would be destroyed. Finally, they were asked to sign an informed consent. As a result of the recruiting process, the Scrum Masters and customer representative of 18 projects agreed to participate.

For each project, we collected data regarding the most recently finished Sprint. The questionnaire respondents had an average of 9.8 years of experience in software projects; six years with Scrum. The projects, on average, were composed of 6.7 members, including developers and testers. We also collected data regarding the progress of each project, in which the progress was calculated by dividing the number of executed sprints by the number of planned sprints.

Out of the 18 projects, ten were already finished, three with more than 85% of progress, one with 58%, and the remaining one with approximately 50% of progress. To better contextualize the projects, we collected data regarding the size of their codebase, measured in lines of code (see Figure 2).

B. Predictive Accuracy

Table II shows the answers of all respondents given the variable “Customer satisfaction” and the calculated results of the model. The values in bold represent the states with a higher calculated probability for the given project. In Table II, we used a color code to interpret the data given the face value by marking with the color green the correct predictions and red the incorrect ones. As a result, for the face value, the models’ predictions were correct for 14 of the 18 projects, resulting in 78% of accuracy, which is a reasonable accuracy. The average Brier score was 0.16. Since the average Brier score was close to 0, it means that Kaizen’s predictions were good [3].

The process to adopt Kaizen in this study was not ideal because of the participants’ availability. For instance, we believe that using multiple questions or objective measures as data sources for the variable factors could have improved the answers’ internal consistency and reduced bias. Given this, we believe that Kaizen’s predictions were satisfactory. We interviewed the respondents of the projects for which the predictions were incorrect, given the face value (projects 2, 8, 12, and 13). For all cases, they argued that their projects were in the product discovery phase and that deliverables were proofs of concepts or MVPs. Thus, the quality rigor was lower than usual. We believe that the incorrect predictions were caused by the lack of proper tailoring of Kaizen, given the company’s context.

A limitation of our analysis for Kaizen’s predictive accuracy is that out of the 18 projects, 16 had an expected Customer Satisfaction value of High. Ideally, from our study’s perspective, we would have a more uniform distribution of the expected values to test Kaizen with diverse situations. However, when collecting data in the real world, such inconsistencies with the ideal scenarios might happen.

---

\(^3\)https://doi.org/10.5281/zenodo.4604001
\(^4\)https://forms.gle/9vtsdNJaC8vLjQfQA

---

![Fig. 2. Distribution of code size for the projects.](https://via.placeholder.com/200)
Given our results, one could argue that if we had just guessed that Customer Satisfaction was High, we would have better accuracy, considering face value (i.e., 16 out of 18). Even though this is true, we believe that this does not mean that such a guess would be better than Kaizen’s predictions in general. In previous studies [13], we have demonstrated Kaizen’s capability of accurately predicting positive and not positive Customer Satisfaction given diverse situations. Clearly, only guessing a High Customer Satisfaction is not enough for diverse scenarios. Thus, such an argument is supported only by the coincidence of the data collected in this study being highly skewed toward High Customer Satisfaction. Consequently, we believe that the results mentioned above for Kaizen’s predictive accuracy is a positive indicator of its reliability, complementing our past results [14], [13], [12].

C. Implications for research and practice

This section discusses our study’s implications for research and practice. For research, our results reinforce Kaizen’s construct validity by complementing our past studies [14], [13], [12] and presenting its predictive accuracy in an industrial case. On this note, it is worthy of mentioning that Kaizen is mainly based on subjective measures. Thus, our results open the opportunity for research to investigate valid quantitative measures that would reduce the subjectivity and effort of using it. We believe that a promising way forward is to analyze how to connect its nodes into data collected by tools used by agile teams, including SonarQube and project tracking software. In other words, Kaizen can be used as a reference to kickoff software measurement programs for Scrum teams focusing on optimizing customer satisfaction.

As previously discussed, Kaizen only focuses on one of the four dimensions of Evidence-Based Management’s Key-Value Areas. Since we have had positive experiences with Kaizen using customer satisfaction as the reference for optimizing the Scrum team’s performance, we believe that opportunities arise in exploring other Key-Value Areas.

Further, we made available Kaizen’s model, and we encourage the research community to investigate its use within a broader context. In terms of industry adoption, we believe that two factors are hindering it. First, there is a need for detailed adoption guidelines and use cases of how Scrum teams can use Kaizen to improve their workstreams. Second, Kaizen relies on AgenaRisk. Thus, there is a need to develop an independent tool that connects with diverse data sources, including the tools used by Scrum teams.

We believe that Kaizen could help immature teams to adopt Scrum by assisting them in detecting deviations or anti-patterns. However, Scrum is a framework, and it does not prescribe tactics since they are context-sensitive. Thus, Kaizen does not inform users about how each of the model’s factors should be fulfilled. Even though this characteristic promotes adoption flexibility, it also hinders its usefulness since it does not guide the users on how to solve the detected problems, for instance, by suggesting using Story Points to estimate Product Backlog items. A line of research is to complement Kaizen’s current version to model such tactics using the concept of causal intervention [2]. Such a model could be the basis for developing information retrieval or recommender systems to help teams define process improvement action points.

Concerning the implications for industrial practice, we have presented in this paper results the validate Kaizen, which is a tool that can help them, at the project level, by assisting the team on risk management, and also at the organizational level by assisting on knowledge-based process improvement and deployment of a software measurement program.
D. Threats to validity

This section discusses this study’s threats to validity following the classification presented by [16].

Construct validity: we used a 3-point ordinal scale for the factors, which are subjective by nature and susceptible to cognitive bias, the illusion of communication, and invalid inferences [6]. Despite this, similar approaches have been used in other studies with positive results [5], [8], [14], [10].

Reliability: since the respondents of the questionnaire are providing data for their teams, the data might be biased. We minimized this potential effect on our results by guaranteeing the participants that the data would remain anonymous and that their performance was not at stake. Further, we pre-tested the questionnaire to avoid the risk of having respondents misinterpret the questions.

Internal validity: we used a selective sampling approach, in which we collected data from one software development company. Further, the participants from the 18 volunteered to participate in the study during our recruiting process within the company. Finally, we gave the participants one week to answer the questionnaire on their best availability to avoid answering it in a rush.

External validity: the questionnaire defined to operationalize the usage of the causal model is company-specific. Therefore, the results are bound by the company’s context in which the study was performed. Despite this, they might apply to other companies that use Scrum to manage software development projects, even though it is expected that companies that adopt Kaizen would need to tailor it to their context.

VI. Conclusions

This paper presented the results of an empirical study to evaluate Kaizen [14], [13], [12] with data collected from 18 projects from one software development company.

A limitation of this study is that out of the 18 projects for which we collected data, 16 expected a High value for our “target variable” (i.e., Customer satisfaction). Ideally, we would have collected data better distributed for the remaining possible states. However, we believe this is a natural risk of collected real-world data. We believe that this aspect does not hinder Kaizen’s evaluation because it has been validated with other datasets previously [14], [13], [12].

Further, this paper presented how to adopt Kaizen by defining a questionnaire as its data source. However, in future work, we will detail an adoption guideline for Kaizen and present use cases of how it can support Scrum teams to identify process improvement opportunities and define action points.

Kaizen’s industry adoption potential is currently limited from an operational perspective because it relies on AgeNaRisk, making it challenging to input data into the model since it is a manual endeavor. Currently, we are working on a tool that implements the required algorithms for Bayesian network inference and enables us to connect external data sources to variables in the model using REST-based endpoints. This feature enables, for instance, connecting the variable Increment with metrics collected on SonarQube [17]. Furthermore, it enables the registration of corrective (or preventive) actions coupled with the factors. As a result, a knowledge base could be created for the organization and used by project managers for data-driven risk management through information retrieval, case-based reasoning, or recommender systems. Additionally, we intend to explore evolving the model through the use of dynamic Bayesian networks to handle the iterative nature of Scrum.

REFERENCES

Towards Automatically Generating Release Notes using Extractive Summarization Technique

Sristy Sumana Nath  
Computer Science  
University of Saskatchewan  
Saskatoon, Saskatchewan  
Email: sristy.sumana@usask.ca

Banani Roy  
Computer Science  
University of Saskatchewan  
Saskatoon, Saskatchewan  
Email: banani.roy@usask.ca

Abstract—Release notes are admitted as an essential document by practitioners. They contain the summary of the source code changes for the software releases, such as issue fixes, added new features, and performance improvements. Manually producing release notes is a time-consuming and challenging task. For that reason, sometimes developers neglect to write release notes. For example, we collect data from GitHub with over 1,900 releases, among them 37% of the release notes are empty. We propose an automatic generate release notes approach based on the commit messages and merge pull-request (PR) titles to mitigate this problem. We implement one of the popular extractive text summarization techniques, i.e., the TextRank algorithm. However, accurate keyword extraction is a vital issue in text processing. The keyword matching and topic extraction process of the TextRank algorithm ignores the semantic similarity among texts. To improve the keyword extraction method, we integrate the GloVe word embedding technique with TextRank. We develop a dataset with 1,213 release notes (after null filtering) and evaluate the generated release notes through the ROUGE metric and human evaluation. We also compare the performance of our technique with another popular extractive algorithm, latent semantic analysis (LSA). Our evaluation results show that the improved TextRank method outperforms LSA.

Index Terms—Software release notes, Extractive text summarization, Software maintenance and documentation

I. INTRODUCTION

Software release is a way to deliver the software package, with release notes, of the stable version to the end-users. Release notes are essential documentation in software development that contains a set of project activities, e.g., issue fixes, improvements and new features that have been implemented to a specific release [18]. Different stakeholders (e.g., software development teams and external users) might benefit from release notes [13], [19]. For example, the development team members (project manager, team lead, developer, tester) use them to learn what has changed in the source code to solve issues or integrate new features [13]. Similarly, integrators, who are using a library in their code, use the library release notes to decide whether such a library should be upgraded to the latest release [19]. End-users and clients read the release notes to decide whether it would be worthwhile to upgrade to the latest software (e.g., application software or tool) version [5]. Besides, release notes serve as valuable resources to generate software documentation (e.g., software release report) submitted to the client. Moreover, practitioners use release notes in requirements engineering, software programming, software debugging and testing phase [4].

Generally, more than one developer is working on a single software project [5]. Generating release notes of a project by an individual is challenging as it is not feasible to know all fixed issues or the integrated new features of the project. Therefore, primary release note producers collect information from contributors to produce the release notes manually [6]. A few studies have proposed the generation of release notes - some of them are semi-automated [13], and some are automated techniques [5], [19] using manually-defined templates. The automated technique ARENA generates release notes by extracting source code change of Java projects and using some manually pre-defined templates based on the source code change type [19] (example in Figure 1). The primary limitation of this method is that it fails if the source code change or code statement is not matched with the pre-defined data, this approach cannot generate a sentence. Besides, this technique is only helpful for Java-based software. In an empirical study [6], 108 GitHub contributors and 206 IT professionals were surveyed for production and usage of release notes. The survey results reveal that none of the participants used ARENA for producing release notes. Whereas the existing approach is not suitable to use; therefore, this is one of the motivations of our study. On the other hand, the semi-automated technique [13] filters good quality commit messages between two releases and appends them all to suggest producing release notes. Appending the commit messages of interrelated releases is not a feasible solution to generate release notes because it generates a lengthy release note. The audience wants to read precise release notes [6]. Some text management tools (e.g., issue trackers) are used to keep track of changes for producing and managing release notes, still need some considerable manual effort to create the release notes from these sources [6].

Developing new tools could greatly help to improve release notes production and usage [6]. As discussed above, there is a great lack of automatic release notes generation tools, which motivates us to develop an automated release notes generation technique. To work collaboratively, developers upload the
source code in social coding platforms, e.g., GitHub, and can track the changes of the projects. A release is a collection of several commits, and commits can be regular commits and merge commits. Table I represents a release note of Laravel project with the related commit messages and merge pull-request titles. Therefore, commit messages and merge pull-request titles are valuable artifacts to generate automated release notes.

We prepare a new dataset that contains the release information (e.g., release notes, release date) with their commit messages and merge pull-request titles of 13 projects from GitHub. After data filtering and pre-processing, our prepared dataset contains over 1,200 release data (described in Section III). In our study, we regard the combination of commit messages and merge pull-requests as an input source and release notes as the summary of the input sources. We apply the TextRank algorithm; however, traditional TextRank algorithm ignores the semantic similarities. We integrate word embedding technique GloVe (Global Vectors for Word Representations) [22] to solve this problem and analyzed its effectiveness (described in Section V-C).

To evaluate the automatically generated release notes, we also implemented the Latent Semantic Analysis (LSA) text summarization technique. We evaluated automated release notes on the dataset using Recall-Oriented Understudy for Gisting Evaluation (ROUGE) [14] and the F1-score obtained with ROUGE-1, ROUGE-2, and ROUGE-L is 31.74%, 18.53%, and 26.90%, respectively. We also conduct a human evaluation to assess the quality of the generated release notes, which shows that our approach performs significantly better than the LSA and can generate more high-quality release notes.

In summary, our contributions are:

- We develop a dataset with over 1,200 releases and their commit messages and merge pull-request titles from GitHub for release notes generation task.
- We implement the TextRank algorithm and integrate the GloVe word embedding technique to generate release notes for software release from their commit messages.
- We evaluate this approach on the dataset using the ROUGE metric and human evaluation.

We describe the research questions and usage scenarios in Section II and data collection process in Section III. We elaborate our proposed approach in Section IV and present the evaluation results in Section V. Section VI and VII discusses threats to validity and the related work. We conclude this paper with some future plans in Section VIII.

II. STUDY DESIGN

Our study aims to answer the following two research questions:

**RQ1: What is the vital information that needs to include in the release notes?** We investigate in the GitHub projects' repositories to understand the contents of the release note. We find that for the most cases developers prepare a list of software changes of the current version from the last previous version. The list of software changes comes from commits and merge pull requests (Table I). For reason, we collect the commit messages and merge pull requests titles as a input source to generate automated release notes.

**RQ2: What is the efficient way to generate release notes?** As per our previous discussion, it is clear that no appropriate tool exists for automated release notes generation in the software development practice. To produce concise release notes, we apply text summarization techniques. We implement the TextRank summarization algorithm by integrating the GloVe model. Using the GloVe model, we can resolve the semantic similarity issue of the traditional TextRank algorithm (detail describe in Section IV). For evaluation, we have used the evaluation metric ROUGE to assess the text quality of the automated summary.

1) **Usage Scenarios:** The release notes may help developers and users to capture a summary of the latest release. Our approach can help developers to write the good quality of release notes. The usage scenarios are as follows:

- It is challenging for developers to keep in mind all the changes for the upcoming release. Therefore, our approach can assist developers in producing release notes.

https://github.com/laravel/laravel/releases/tag/v8.4.2
• Our approach can help to replace the existing empty release notes in GitHub.

![Fig. 2: Sample Release Note](https://github.com/django/django/releases)

2) Problem Formulation: Inspired from the example shown in Table I, we observe the production of the release notes as a text summarization task with the combination of the commit messages and merge PR titles in the release as the article and the release notes as the summary. Therefore, the problem is formulated as follows:

Given a source sequence $S_{seq} = (x_1, x_2, ..., x_n)$, $n$th sentences in source sequence. An extractive summarizer aims to produce a summary $RN_{gen}$ by selecting $m$ sentences from $S_{seq}$, where $m$ is the length of existing release notes $RN_{ref}$. We use a scoring function $f()$ to generate a score for every sentence of $S_{seq}$ and then select top-ranked $m$ sentences to produce an automated summary.

III. DATASET

A. Data Preparation:

In our study, we need to develop a dataset of release notes with commit messages and pull-request titles of the GitHub projects. To eliminate the trivial projects, we defined three criteria for project selection from GitHub projects: (i) the project is active (i.e., the repository of the project is being updated); (ii) the number of release notes in the project is more than 40; and (iii) the project have more than 8,000 stars. Generally, GitHub projects are classified into six domains [8]: (1) Application software, e.g., browsers, text editors; (2) System software, e.g., operating systems, (3) Web libraries and frameworks, e.g., Web API, (4) Non-web libraries and frameworks, e.g., android framework, (5) Software tools, like IDEs, and compilers; (6) Documentation, like documentation, tutorials.

We extract 1,924 release data (see in Figure 2), e.g., release note, hash, release data and tag name, of five domains except Documentation repositories of 13 projects (among them 6 Java, 3 Python and 4 PHP projects). We find 711 (37%) release have empty release notes and develop a dataset with 1,213 release data. After that, we extracted the commit messages and merge pull-request titles of the releases. In GitHub, the new commit is linked with the previous commit using the parent, i.e., hash value (shown in Figure 3). We extract all parent hash values between two releases and then extract the commit messages and merge PR titles.

B. Data Pre-processing:

We conduct some text pre-processing techniques in the release notes and source sequences to filter out noise from the text. First, we eliminate the empty release notes. Then, we remove the HTML tags from the extracted release notes. Then, we split the text into sentences using NLTK (Natural Language Toolkit [7]) and delete the a) url, b) reference number (e.g., "#123"), c) signature, e.g., signed-off-by or co-authored-by, d) '@name', e) markdown headlines, e.g., ###Added, by identifying through regular expressions. Then, we filter out some trivial commit messages (for example, duplicate commit messages in same releases, 'merge pull request/branch', ‘update .gitattributes’ and so on).

IV. APPROACH

Automatic text summarization process produces a summary by reducing the text of a document. Text summarization approaches can be categorized into two types, extractive and abstractive summary. Extractive summary is created by extracting the phrases or sentences from the input sources and abstractive summarization process produces new words by understanding the main content of the input sources. We use extractive techniques in our study. Usually, extractive summarization processes follow three steps to generate summary: (a) intermediate representation of the document, (b) scoring sentences and (c) strategies for selection of summary sentences. The procedure of our approach is described in Figure 5.

![Fig. 3: Commit](https://github.com/django/django/releases)

![Fig. 4: Data Collection Process](https://github.com/sristysumana/SEKE2021Paper119)
A. TextRank Algorithm

TextRank is an unsupervised graph-based ranking algorithm, and it is widely used in keyword extraction and text summarization. In general, graph-based ranking algorithms decide the importance of a vertex within a graph based on global information iteratively drawn from the entire graph. The core idea of the TextRank algorithm is to split the whole document into sentences. These sentences are used as nodes, and the weight between the nodes is used as edges. The weight set of edges, and the weight of elements can be defined as:

$$Sim(S_i, S_j) = \frac{|\{t_k | t_k \in S_i \& t_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)}$$

(1)

According to the $Sim(S_i, S_j)$ value, an similarity matrix $SM_{n \times n}$ is constructed between nodes. The weight of elements on the diagonal of $SM_{n \times n}$ are all 1. Based on similarity matrix, the TextRank algorithm generates the importance or score of vertex $S_i$ using following equation,

$$S_c(S_i) = (1 - d) + d \times \sum_{S_j \in In(S_i)} \frac{w_{ji}}{\sum_{S_k \in Out(S_i)} w_{jk}} Sc(S_j)$$

(2)

Here, $S_c(S_i)$ is the ranking score of the node $S_i$, $d$ is the damping factor that represent the probability of the current node jumping to any other node, and at the same time can enable the weight to be transferred to the converged stably. This co-efficient value can be set between 0 to 1, however $d$ is usually set to 0.85 [17], [21]. $In(S_i)$ is the collection of all nodes pointing to node $S_i$, $Out(S_i)$ is the set of all nodes pointed by node $S_i$. The sum of the right side in the equation (2) indicates the contribution of each adjacent node to the node. The summed numerator $w_{ij}$ represents the degree of similarity between the two nodes $S_i$ and $S_j$ as well as the denominator is a weighted sum. $S_c(S_j)$ represents the weight value of node $S_j$ after the last iteration. After generating the score of all sentences, the TextRank algorithm generates a summary by selecting top-scored sentences.

B. Text Vectorization

Text Vectorization is the process of transforming text into numerical representation. Several ways are existing for the text vectorization, like, bag-of-words (BoWs), term frequency-inverse document frequency (TF-IDF), global vector for word representation (GloVe). The traditional TextRank algorithm converts the document into BoWs vector representation, which is an unordered collection of word counts [17] and the size of this vector is equal to the number of words in the vocabulary. The main issue of the BoW models is that if a sentence comes with the new words, then the vocabulary size would increase as well as the length of the vectors would increase. For that reason, the vector representation of the BoW will be a sparse matrix. Sparse representations are harder to model both for computational reasons and informational reasons as well as required a huge amount of training data. To resolve this issue, the modified TextRank algorithm [16] integrates the TF-IDF algorithm. TF-IDF model reflects the importance of a word to a document in a collection or corpus. It can keep the relevant words score in the sentences and also considers the different lengths of the sentence in the document. TF measures the frequency of the word in each document in the corpus and IDF calculates the weight of rare words across all documents in the corpus. The formula of $TF - IDF(w_i)$ is as follows:

$$TF(w_i) = \frac{f_{S_j}(w_i)}{f_{S_j}(w)}$$

$$IDF(w_i) = \log \frac{N_s}{f_S(w_i)}$$

$$TF - IDF(w_i) = TF(w_i) \times IDF(w_i)$$

where $f_{S_j}(w_i)$ represents the number of repetition of the word $w_i$ in the sentence $s_j$, $f_{S_j}(w)$ is the total number of words $w$ in the sentence $s_j$, $N_s$ represents the total number of sentences in the corpus; $f_S(w_i)$ represents the number of sentences containing the word $w_i$. 

![Fig. 5: Overview of the methodology](image-url)
TABLE II: ROUGE scores

<table>
<thead>
<tr>
<th>Approach</th>
<th>ROUGE 1</th>
<th></th>
<th></th>
<th>ROUGE 2</th>
<th></th>
<th></th>
<th>ROUGE-L</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>Precision</td>
<td>F1-Score</td>
<td>Recall</td>
<td>Precision</td>
<td>F1-Score</td>
<td>Recall</td>
<td>Precision</td>
<td>F1-Score</td>
</tr>
<tr>
<td>LSA</td>
<td>25.65%</td>
<td>30.28%</td>
<td>29.44%</td>
<td>14.89%</td>
<td>14.35%</td>
<td>14.61%</td>
<td>23.54%</td>
<td>25.45%</td>
<td>24.45%</td>
</tr>
<tr>
<td>TextRank+GloVe</td>
<td>30.23%</td>
<td>33.42%</td>
<td>31.74%</td>
<td>17.29%</td>
<td>19.97%</td>
<td>18.53%</td>
<td>24.29%</td>
<td>30.15%</td>
<td>26.90%</td>
</tr>
</tbody>
</table>

Still, capturing semantic similarity between the sentences is the issue of the TextRank algorithm. We observe that generating automated release notes needs to consider semantic similarity among the input sources (e.g., commit messages). Therefore, we integrate the GloVe word embedding model with the TextRank algorithm, which can improve keyword extraction performance by enhancing the semantic representations of documents. GloVe approach generates vector representation by calculating the co-occurrence matrix of each word based on the frequency of word and co-occurrence counts. It primarily gives information about the frequency of two words $W_i$ and $W_j$ appear together in the huge corpus. To store this information, GloVe model generates co-occurrence matrix $X_{wi}$, each entry of which corresponds to the number of times word $j$ occurs in the context of word $i$. As the consequence:

$$P_{WiWj} = P(W_j|W_i) = \frac{X_{WiWj}}{X_{Wi}}$$

is the probability that word with index $j$ occurs in the context of word $i$. This co-occurrence probability matrix gives a vector space with meaningful sub-structures.

V. EVALUATION RESULT & DISCUSSION

To compare the quality of generated release notes, we implement another text summarization technique named Latent Semantic Analysis (LSA) [20]. It is an unsupervised approach, not required to train the model. LSA extracts hidden semantic structures of words and sentences. An algebraic method, Singular Value Decomposition (SVD), is used to find out the interrelations between sentences and words.

A. Evaluation Metric

We evaluate the automated release notes with the ROUGE metric [14], which has been shown the quality of summarized text by high correlation with human assessments. Specifically, we use ROUGE-N (N=1,2) and ROUGE-L, which are widely used to evaluate text summarization systems. The score of ROUGE-N is based on comparing n-grams in the generated and the original summary. The recall, precision and F1-score for ROUGE-N are calculated as follows:

$$\text{Recall}_n = \frac{\text{Count}_{gram_n}(RN_{ref}, RN_{gen})}{\text{Count}_{gram_n}(RN_{ref})}$$

$$\text{Precision}_n = \frac{\text{Count}_{gram_n}(RN_{ref}, RN_{gen})}{\text{Count}_{gram_n}(RN_{gen})}$$

$$F_1^n = 2 \frac{\text{Recall}_n \times \text{Precision}_n}{\text{Recall}_n + \text{Precision}_n}$$

where, $\text{Count}_{gram_n}(RN_{ref}, RN_{gen})$ calculates the number of overlapping n-grams found in both the original and the generated text. The precision, recall and F1-score for ROUGE-L are similar with those for ROUGE-N, but instead of n-grams, they are calculated using the longest common subsequences between generated descriptions and reference descriptions [14]. ROUGE is usually reported as a percentage value between 0 and 100. We obtained ROUGE scores using the ‘rouge-score’ package [1] with Porter stemmer enabled.

B. Experiment Settings

We use pandas, Python library, for data manipulation and analysis. In text pre-processing phase, we use nltk library for tokenization and removing stop-words as well as write some regular expressions in the code to identify and delete the reference code, markdown, etc. from the text. In this study, we generate extractive summary for producing automated release notes. Therefore, we count the number of sentences ($n$) of reference text and set $n$ is a parameter in the summarization technique. In order that, the sentence length of the reference and the generated release notes remain the same. For implementation, we use several library like, gensim [23], sumy [2], sklearn [9], networkx [11].

C. Analysis and Discussion

1) The Effectiveness of Our Approach: To investigate the effectiveness of our approach (i.e., TextRank+GloVE), we
evaluate the automated release notes with reference text on our dataset and the results are shown in Table II. TextRank+GloVe model has a higher precision and F1 score than the LSA model for each ROUGE metric. The improvements compared with LSA in terms of the three F1 scores are 2.30, 3.91 and 2.44 points, respectively. These results indicate that compared with the two models, our approach can capture the key points of a release more precisely. The TextRank approach integrates the GloVe model, for that reason, this technique can consider semantic similarity between two sentences. On the other hand, sentence scoring and selection process of LSA depends on co-occurrence of words. For example, LSA approach extract “suppress undeliverable exception handling in tests” and “updating suppress undeliverable exception to have a named inner class instead of an anonymous inner class.” in Table III. TextRank+GloVe can handle this situation by generating co-occurrence matrix of words based on the context.

2) The Effects of Main Components: In this study, we generate release notes based on the TextRank approach by integrating GloVe word embedding technique. We compare the automated release notes between TextRank+GloVe and TextRank+TF-IDF with the reference release notes (Table IV). TextRank+TF-IDF model extracts keywords based on the term frequency (TF) and inverse document frequency (IDF) and generate sentence score based on the high TF-IDF value. In release note example II (Table IV), TextRank+TF-IDF model identifies error, missing, components as top keywords and selects top three scored sentences which have these keywords more than the other sentences. On the other hand, TextRank+GloVe extract better result than the TextRank+TF-IDF model. Because GloVe is a very powerful word vector learning technique which does not only rely on local context information of words, but also incorporates global word co-occurrence to obtain word vectors. For example, fix...replay and fix...switchmap are co-occurred in two sentences of source sequence in Table IV. Therefore, our model selects good sentences than the TextRank+TF-IDF. Table V shows the result by comparing the generated release notes by TextRank+TF-IDF and TextRank+GloVe model.

3) Where Does Our Approach Fail: We carefully inspected the release notes where our approach does not obtain good F1 scores. We find that our approach usually performs badly because it presents the top ranking sentences and cannot generate new sentences. The common case is that the source sequence contains several bug fix commit messages, however developers described ‘This release fixes few minor issues’ in the release notes to avoid a lengthy release notes. But our approach extracts a high ranked sentences from the source sequence. This issue is very common, hence it gets low ROUGE scores. Table VI represents a example where our approach fails. Additionally, the length of the generated text is a crucial matter for the poor score, because every sentence has a different word size. In our implementation, we only consider the total number of sentences, the ROUGE score would have been better if we had used the number of words to generate the summaries. However, in the extractive summary techniques, the word number cannot be used as a parameter. To reduce these limitations, we will implement sequence-to-sequence model to produce high-quality release notes in future and train the model.

D. Human Evaluation

In this section, we invite 16 software engineers and get response from 13 participants to assess the quality of the generated release notes by LSA and TextRank algorithms. Automatic evaluation metric, ROUGE, calculates the textual similarity between the existing and the generated release notes, while the human study can evaluate the semantic similarity between them.

1) Procedure: We also conduct a human evaluation to investigate our approach’s effectiveness. We invited 13 human evaluators to assess the quality of the automated release notes. All of them are software developers with 3-8 years of experience and use GitHub. We randomly select 10 release notes from the dataset and design an online survey. For each release, we show its original release notes followed by the two automated release notes generated by the TextRank+GloVe and LSA approaches to the evaluators. The two generated release notes are randomly ordered. Participants also have no idea about how these approaches work, so they cannot figure out which description is generated by which approach. Every evaluator provided a score from 1 to 7 to measure the semantic similarity between the generated release notes and

---

### TABLE IV: Release Note Example II

<table>
<thead>
<tr>
<th>Source Sequence:</th>
<th>Reference:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.x: add error handling section to observable.blockingfirst documentation</td>
<td>fix switchmap not canceling properly during onnext-cancel races.</td>
</tr>
<tr>
<td>3.x: add missing coverage, update unused/inconsistent ops components</td>
<td>fix error occur when missing version tag from single.concatmaps</td>
</tr>
<tr>
<td>error occur when missing version tag from single.concatmaps</td>
<td>3.x: fix incorrect sync-fusion of switchmap and error management</td>
</tr>
<tr>
<td>add more tags those are missing previously</td>
<td>3.x: fix reply not resetting when the connection is disposed</td>
</tr>
<tr>
<td>update 2.x maintenance date, include 3.0 wiki</td>
<td>3.x: fix excess item retention in the other replay components</td>
</tr>
<tr>
<td>update readme.md</td>
<td>3.x: fix excess item retention in the other replay components</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>TextRank + TF-IDF:</th>
<th>TextRank + GloVe:</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.x: add missing coverage, update unused/inconsistent ops components</td>
<td>3.x: fix switchmap not canceling properly during onnext-cancel races.</td>
</tr>
<tr>
<td>error occur when missing version tag from single.concatmaps</td>
<td>3.x: fix error occur when missing version tag from single.concatmaps</td>
</tr>
<tr>
<td>3.x: fix incorrect sync-fusion of switchmap and error management</td>
<td>3.x: fix reply not resetting when the connection is disposed</td>
</tr>
<tr>
<td>3.x: fix excess item retention in the other replay components</td>
<td>3.x: fix excess item retention in the other replay components</td>
</tr>
<tr>
<td>3.x: fix switchmap not canceling properly during onnext-cancel races.</td>
<td>3.x: add error handling section to observable.blockingfirst documentation</td>
</tr>
</tbody>
</table>

---

246
TABLE V: Effectiveness of the GloVe model

<table>
<thead>
<tr>
<th>Approach</th>
<th>ROUGE 1 Recall</th>
<th>Precision</th>
<th>F1-Score</th>
<th>ROUGE 2 Recall</th>
<th>Precision</th>
<th>F1-Score</th>
<th>ROUGE-L Recall</th>
<th>Precision</th>
<th>F1-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextRank+TF-IDF</td>
<td>26.51%</td>
<td>28.49%</td>
<td>27.83%</td>
<td>13.86%</td>
<td>14.08%</td>
<td>13.64%</td>
<td>22.06%</td>
<td>24.49%</td>
<td>23.51%</td>
</tr>
<tr>
<td>TextRank+GloVe</td>
<td>30.23%</td>
<td>33.42%</td>
<td>31.74%</td>
<td>17.29%</td>
<td>19.97%</td>
<td>18.53%</td>
<td>24.29%</td>
<td>30.15%</td>
<td>26.90%</td>
</tr>
</tbody>
</table>

TABLE VI: Release Note Example III

<table>
<thead>
<tr>
<th>Source Sequence:</th>
</tr>
</thead>
<tbody>
<tr>
<td>fixes failed to disable slave database and fixes unit test errors</td>
</tr>
<tr>
<td>fix oracle connection getschema()</td>
</tr>
<tr>
<td>fix shadow order</td>
</tr>
<tr>
<td>fix test cases and update encrypt strategy name</td>
</tr>
<tr>
<td>fix metric image path</td>
</tr>
<tr>
<td>fix integration test for select_with_case_expression</td>
</tr>
<tr>
<td>Reference:</td>
</tr>
<tr>
<td>this release fixes few minor issues</td>
</tr>
<tr>
<td>TextRank+GloVe:</td>
</tr>
<tr>
<td>fixes failed to disable slave database and fixes unit test errors</td>
</tr>
</tbody>
</table>

25 20 15 10 5 0

Fig. 6: Score of Human Evaluation

VI. THREAT TO VALIDITY

One threat to validity is that our dataset is built from Java, Python and PHP projects in GitHub repositories. Therefore, it may not represent all programming languages. Besides, our approach takes commit messages and merge pull-request titles as input; hence can also be applied to projects of other programming languages.

Another threat to validity is that the non-summary information, such as acknowledgment of contributors, installation command for the new release, in release notes may affect the effectiveness of our approach. Release notes are free-form text, and we cannot guarantee their quality and content. We try to mitigate this threat by using a set of heuristic rules and manual analysis to filter out non-summary information when pre-processing. However, it is hard to process the patterns of all non-summary information. For future work, we will focus on data pre-processing for further improvements.

Another threat is we cannot compare the generated release notes from ARENA and our approach because ARENA extracts all issues from JIRA to generate release notes for Java projects and generate lengthy release notes. Moreover, from survey study, no participants have adopted ARENA for producing release notes [6].

There is also a threat related to our human evaluation. We cannot guarantee that each score assigned to every release note is fair. To mitigate this threat, we invite the professional developers to our survey who have experience in GitHub and producing release notes.

VII. RELATED STUDY

Several empirical studies focused on the usage of release notes. Abebe et al. [3] identify nine important factors (e.g., issue priority and type) in explaining the likelihood of an issue being included in release notes. Tingting et al. [6] analyzed 32,425 release notes and classified the content into eight categories, e.g., issues fixed, new features, and internal changes. Among them, issues fixed (79.3% of the release notes) and new features (55.1% of the release notes) are the most documented categories. In GitHub, developers push commits or send pull requests in a separate branch to resolve the issues. For that reason, we propose an automated release notes generation technique based on the commit messages and merge PR titles.

Some other studies aimed to generate automated release notes. Moreno et al. [19] propose ARENA tool to produce release notes for Java projects. ARENA extracts the source code changes from GitHub and collect issues from JIRA using issue-commit linker. Then it prepare a list of issues and generate change description using predefined text templates. The example is shown in Figure 1. Similarly, Ali et al. applied the same technique to generate automated RNs for Node.js projects [5]. Klepper et al. [13] proposed a semi-automatic
approach by collecting information from the build server, issue tracker and version control system. Then, release manager can edit the this list before publishing. However, we implemented full-automatic approach to produce release notes and pre-defined text templates is not required in our study.

For automated software documentation, researchers studies different types of software artifacts, such as commit messages generation [10], [12] and automated pull-requests description [15]. For example, Liu et al. [15] propose an automatic approach to generate PR descriptions based on the commit messages and the source code comments. On the other hand, release combines plenty of commits and pull requests. Our work aims to generate release notes from commit messages and pull-request titles using an extractive method.

VIII. CONCLUSION & FUTURE WORK

This paper aims to generate release notes for software releases automatically. In our work, we apply the extractive text summarization technique, i.e., TextRank, to produce automated release notes by selecting top-ranking sentences from their commit messages and merge PR titles. The main novelty of this work is that we do not use any pre-defined template like ARENA [19]. Our approach is language-independent; oppositely, ARENA is used for Java projects. Moreover, we integrate GloVe to convert the text into embeddings, which helps to extract keywords than the traditional approach. We evaluate the generated release notes using the ROUGE metric and conduct a human evaluation to check the effectiveness of the automated release note generation technique.

In future, we plan to apply a sequence-to-sequence model to improve the quality of release notes. We also plan to improve our approach by involving additional related software artifacts as input. For example, by taking git diff, relevant content from bug reports as input and text summarization models may be able to infer the implementation details and the motivation of a software release. Moreover, we plan to improve the structure of release notes. We will categorize, e.g., bug fixes, improvements, new features, the content of release notes for better representation to release note users.

ACKNOWLEDGMENT

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC), and by two Canada First Research Excellence Fund (CFREF) grants coordinated by the Global Institute for Food Security (GIFS) and the Global Institute for Water Security (GIWS).

REFERENCES

[1] Rouge-score 0.0.4. https://pypi.org/project/rouge-score/.
Understanding the Impact of COVID-19 on Github Developers: A Preliminary Study

Liu Wang1, Ruiqing Li2, Jiaxin Zhu3, Guangdong Bai2, Weihang Su1, Haoyu Wang1
1 Beijing University of Posts and Telecommunications, Beijing, China
2 The University of Queensland, Australia 3 Institute of Software, Chinese Academy of Sciences, China

Abstract—The ongoing COVID-19 pandemic has impact almost every aspect of human lives profoundly. This paper investigates the impact of COVID-19 on the activity and contribution of open source software (OSS) developers. Specifically, we make great efforts to harvest the information of all the developers (over 25 million) on GitHub and their contribution activities. With such a large-scale dataset, we perform analysis from four perspectives, including the overall ecosystem level, country level, organization level and developer level, to characterize the impact of COVID-19 on the OSS community. We have revealed a number of interesting observations and trends, which are crucial to understanding the OSS contributors and supporting the collaboration to combat global crisis like COVID-19.

Index Terms—COVID-19, Github, Developer Contribution, Open Source Software

I. INTRODUCTION

The COVID-19 pandemic has changed people’s lives in many ways. In this paper, we intend to take a look at the GitHub developers and their contributions during the pandemic, and share our preliminary results.

GitHub developers can make many kinds of contributions. Commits, pull-requests and issue reports are the most commonly studied contributions in literature. In this paper, we consider the comprehensive contributions monitored and recorded by GitHub, which has been seldom analyzed before. In particular, ‘committing to a repository’s default branch or gh-pages branch’, ‘opening an issue’, ‘proposing a pull request’, and ‘submitting a pull request review’ are counted as contributions [1]. GitHub presents a contributions graph for each user in his/her profile page. It tracks the number of daily contributions and visualizes how active one has been on the site, which we believe is a good indicator to measure the activeness of GitHub developers during COVID-19.

Basically, we study the number of contributions per day and the number of developers who make these daily contributions before and during the pandemic. We first make a huge effort to harvest a comprehensive dataset of developers in GitHub and their daily contributions. By the time of November 10, 2020, we have collected 25,761,884 developers in total. Leveraging the dataset, we perform a systematical analysis including four perspectives, i.e., the overall ecosystem level, country level, organization level and developer level, which corresponds to four research questions that drive our study. Our first research question is how the overall numbers change (RQ1).

GitHub developers are from all over the world, and some of them are corporate employees. The severity of the pandemic varies across countries. Different companies also have different strategies to handle the situations. Therefore, we raise the next two research questions, how the numbers change within different countries (RQ2) and different companies (RQ3). The developers are of different activeness, and we also want to see the trends among them (RQ4).

We observe that there is a remarkable increase in developer participation and contribution to GitHub in the early stages of the COVID-19 global explosion. In terms of individual countries, the trends of GitHub developer contributions are tied to the outbreak in the corresponding country. On a company level, the work-from-home setting implemented due to the pandemic seems not to have disrupted software developers’ commitment to GitHub. For developers of various activeness levels, their involvement and contribution to GitHub more or less increased during the COVID-19 outbreak, even for those developers who are very inactive. Our observations suggest that COVID-19 did not pose great challenges to GitHub developers and show that GitHub may have played an important role in COVID-19, tapping into the potential of OSS communities like GitHub to respond to public crises.

II. RELATED WORK

Since its outbreak, COVID-19 has attracted great attention from various research communities. A large number of studies were focused on the medical domain [2]–[4]. In the field of mobile app analysis, Wang et al. [5] conducted a systematic analysis of coronavirus-themed mobile malware and revealed huge potential threats beyond the virus. There is also a number of research on COVID-19 in the software engineering community. For example, Wang et al. [6] presented a large-scale empirical study of COVID-19 themed repositories on GitHub and highlighted the practical and potential value of open source technologies and resources in handling such crisis.

In addition, GitHub developers (users) have been studied from many perspectives in literature. Technical roles of the developers and experts of the open source projects are automatically identified via machine-learning based approaches [7], [8]. Contributions of novice developers are characterized [9], and the effect of developer sentiment on fix-inducing changes is analyzed [10]. Collaboration among GitHub developers is investigated to identify the characteristics favoring innovation in the open source community [11]. Researchers also studied
the motivation behind following others and the influence of popular users on their followers [12]. Besides, some studies have attempted to understand developer productivity at technology companies due to the almost overnight migration of software developers to work from home. For example, Bao et al. [13] presented a case study on Baidu Inc. to investigate the difference of developer productivity between working from home and working onsite. Ford et al. [14] conducted survey studies to understand the benefits as well as challenges of working from home and analyze the factors that have affected developer productivity over time. Ralph et al. [15] performed a questionnaire survey study to investigate the effects of the pandemic on developers’ well-being and productivity.

The researches above only focus on a small number of GitHub developers. Chatziasimidis and Stamelos conducted some measurements and association mining with 100K projects and 10K GitHub users/owners of the projects [16]. Different from those studies, in this paper, we target at all the GitHub developers to understand the impact of COVID-19.

III. DATA COLLECTION

Aimed at all the GitHub developers, a large-scale dataset is retrieved from GitHub. We employ the GitHub search API [17] to find all the GitHub developers who have at least one public repository. Considering the limitation of 1,000 results per search, we have adopted a segmented approach, i.e., narrowing down the results of a single query using some search qualifiers and performing multiple queries to further consolidate the results to collect a complete list of developers. To do this, we filter the developers based on when they joined GitHub with the created qualifier, which takes a date as its parameter with optional time information after the date to search by the hour, minute, and second. As a result, we collect a total of 25,761,884 developers whose GitHub accounts were created from 2007 to 2020. Additionally, we plot the distribution of creation dates for all developers in our dataset in Figure 1. It can be observed that the number of daily new developers overall exhibits a continuous upward trend and has a peak in mid-2019, which in a way showcases the growing market of GitHub over the past decade. The basic information we collect includes id, login, name, location, company, etc. Of which, the location field reveals that these developers are located in almost every country in the world today, and the distribution map of the number of developers per country is shown in Figure 2. The number of developers providing locations is considerably higher in the United States than in other countries. More importantly, for all the developers collected, we capture their daily contributions (e.g., commits, issues and pull requests) from their creation date to November 10, 2020 by crawling the profile pages. The entire data collection involves over 26 million HTTP requests, which is a very time and network resource consuming task.

IV. EMPIRICAL RESULTS

We present the answers to our research questions of the overall ecosystem level, country level, organization level and per-developer level in the following sub-sections.

A. RQ1: Overall Trends

Figure 3 shows the sum of contributions per day in 2019 and 2020 for all developers whose accounts were created before 2019. We can examine the trend of contributions in three stages: (1) Prior to mid-March, i.e., before the global pandemic of COVID-19, the total number of daily contributions exhibits a largely stable trend, with only a slight decline in late-2019 and early-2020, which probably due to the Christmas and New Year holidays. (2) From mid-March to the end of June, the period when COVID-19 is exploding worldwide, and the number of confirmed cases is rising sharply, meanwhile, the daily contribution of developers shows a clear trend of increase, peaking in April and gradually falls back in May. (3) After July, the trend has dropped to flat, but appears to be slightly higher than that in 2019.

We then use Mann-Whitney U test [18], a popular non-parametric test to compare outcomes between two independent groups, to test whether there is a statistically significant difference between developer contributions in 2019 and 2020. Specifically, we generate two samples containing developer contributions in 2019 and 2020, then calculate the test on the samples and print the statistic and p-value (2.43e-35). Typically, if the p-value is below 0.05, the test says there
is enough evidence to reject the null hypothesis and that the samples were likely to have different distributions. Thus this p-value (2.43e-35) strongly suggests that the sample distributions are different, i.e., the developer contributions in 2020 is statistically significantly different from that of 2019, as expected. This observation suggests that the activity of developers is not significantly disrupted by the pandemic. Instead, it has somewhat boosted developers’ engagement in GitHub, especially in the early stages of the global pandemic. In a way, it also reflects that the open source community and collaborative platforms like GitHub may have an important role to play in the face of a public crisis such as COVID-19. Beyond that, it is interesting to see that the trend is cyclical, and the periodicity is one week upon our inspection, which is quite likely related to the weekly work schedule of most employees, even during the pandemic.

Further, we attempt to explore the reasons for the increase of the daily developer contributions in the second stage from two aspects. On the one hand, we count the number of active developers whose contribution value is greater than 0 for each day in 2020, as shown in Figure 4(a). We can observe that the trend is similar to that in Figure 3, with a significant peak-like trend experienced from mid-March to the end of June. This suggests that the number of developers making contributions per day has increased during the second stage, which leads to the trend of Figure 4(a). On the other hand, we calculate the average contribution value of all active developers for each day as displayed in Figure 4(b). It can be seen that the average values are very close, fluctuating from 4.2 to 5.2. Although a slight jump in the average can be seen at the end of March, the increase is too small to cause the significant growth in the second stage. To sum up, there was a significant increase in the number of active developers in GitHub during the period when COVID-19 began to wreak havoc around the world. In particular, regarding the outlier that appears on April 11, 2019 in Figure 3, we tried to explain it by manually checking the number of active developers and the average contribution of active developers during this period. We found that the number of active developers on this day was not particularly high but the average contribution showed a similar outlier, suggesting that some developers may have made a substantial amount of contributions on this day.

B. RQ2: Country-level

As aforementioned, we collect the location information of all developers, while unfortunately 85% of them did not provide their locations. Among the available locations (3,821,170), different formats/representations are used, including city name only, country name only, and both. Thus we collect a list of countries in the world and major cities in each country and use the string matching method to determine the country to which the location belongs. Finally we acquire a total of 3,647,523 (95.5%) valid locations mapping 195 countries. The top three countries in regards to the number of developers are the United States (698,266), India (379,286) and China (300,496). We explore whether the contribution of developers is significantly associated with the corresponding pandemic situation at the national level within the three leading countries. For each country, we collect the daily number of new COVID-19 confirmed cases over time from Google Statistics [19] and place them in charts for comparison, as well as marking the start of lockdown, as shown in Figure 5. It can be seen that all three countries appear to have a rapid rise in daily contributions in the early stage of the outbreak, even though at different rates, e.g., China and India rise significantly, while the US rises slightly.

For the sake of statistical validity, we use the Pearson correlation coefficient to analyze the relationship between total

![Fig. 3. Distribution of the total number of contributions per day for developers whose accounts were created before 2019.](image)

![Fig. 4. The trend of active developers and average contribution per day for developers created before 2020.](image)
contributions and pandemic situation for each country. Since the impact of COVID-19 on developer contributions is mainly observed at the beginning of the outbreak, we calculate the correlation between the number of newly confirmed cases and developer contributions in the one-month period following the lockdown for each country. We find that the developer contributions of the three countries were all positively correlated with their corresponding pandemic growth in varying degrees, with stronger correlations in China ($r=0.35$, $p$-value=0.06) and India ($r=0.29$, $p$-value=0.12), and weaker correlations in the United States ($r=0.04$, $p$-value=0.84). The correlations, though, are not statistically significant (which is speculated due to the volatility of the contributions). Hence, it seems that the growth of developer activity and contributions to GitHub are correlated with the outbreak in the corresponding country.

C. RQ3: Organization-level

COVID-19 has forced companies all over the world to adapt to and embrace remote work—at least for the short term. Large tech employers such as Apple, Google, Facebook and Microsoft are among the first to ramp up remote work plans for many or all of their employees around the globe in March. As aforementioned, we also collect the company information of all developers. There are 1,835,336 (7.1%) developers providing their company information. With these available data, we count the number of developers in major companies around the world, and obtain the top three, i.e., Microsoft (21,296), Google (10,041) and IBM (8,101). Figure 6 shows how the activity and contributions of developers working at these companies have changed in 2020. It can be seen that, as companies move into “work from home” mode in response to COVID-19 at the end of March, developer activity and contributions to GitHub have increased to varying degrees, and although some companies (e.g., Microsoft) have not shown a significant increase, none have decreased. It suggests that companies are likely to rely more on open source communities and collaboration platforms like GitHub for their work-from-home efforts and also implies the potential and usefulness of these platforms in the face of such public emergencies.

D. RQ4: Developer-level

As reported in §IV-A, the number of active developers on GitHub increased significantly during the rise of COVID-19, thus we would like to learn more about what kinds of developers became active during this period. We seek to categorize all the developers into several types based on how active they are on GitHub. First, for each developer, we calculate the percentage of days they are active on GitHub out of the total number of days they have been on GitHub, and find that it is very small for the vast majority of developers, with less than 1% of developers having more than half of the total number of days active. Then, we group all developers (excluding those who have never contributed) into four levels, i.e., very active, moderately active, inactive and very inactive, as shown in Table I. Figure 7 presents the trend of the number of active developers per day for each type of developers separately. As can be seen, the activity of all types of users increased to a greater or lesser extent at the beginning of the COVID-19 global outbreak, with the less active developers increasing considerably, while the more active users showing relatively minor changes, suggesting that a lot of inactive developers got involved in the use of GitHub during this period. Similarly, Figure 8 shows the trend of the number of daily contributions for each type of developers, which is generally very similar to Figure 7. Both charts show that contributions from any type of developers increased as COVID-19 began to explode, and that even among very inactive groups many more people contributed to GitHub during this period.

<table>
<thead>
<tr>
<th>Type</th>
<th>Proportion of Active Days</th>
<th># Developers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Active</td>
<td>[0.3, 1]</td>
<td>28,377</td>
</tr>
<tr>
<td>Moderately Active</td>
<td>[0.1, 0.5]</td>
<td>694,689</td>
</tr>
<tr>
<td>Inactive</td>
<td>[0.01, 0.1]</td>
<td>4,302,035</td>
</tr>
<tr>
<td>Very Inactive</td>
<td>(0, 0.01)</td>
<td>31,838,464</td>
</tr>
</tbody>
</table>

To gain a deeper understanding of developers’ activities, we next explore how the developer contributions evolve over time and try to group the evolution into several patterns. Since inactive users make little or no contribution for most of the time, we only focus on the very active developers (the first type in Table I) and examine the trends of their contributions in the year 2020. For each active developer, we retrieve the contribution he/she made on Github per day over the year, take them as feature vectors and perform L2 regularization [20] to apply the K-Means clustering. We use the Elbow Method to find the optimal value of $k$, which is 3. Thus we group the developer contributions into three clusters. Figure 9 presents
the trends of contributions in each cluster. It can be seen that each cluster captures the weekly work cycle. We next briefly describe the characteristics and examples of different patterns.

Cluster 1 (in red) shows an overall downward trend, gradually declining from March until it stabilizes after August. There are 6,366 (22.4%) active developers whose contribution trends belong to this cluster. This pattern reflects the presence of some active developers whose contributions and activeness have decreased after the outbreak. For example, a developer with the account name ‘najahiiii’ has a contribution trend showing that his daily contribution fluctuates between 7 and 38 in the first three months of 2020 with a daily average contribution value of 12.2, while in the following months it has stayed below 15 with an average daily contribution of 8.

Cluster 2 (in green) indicates an overall upward trend, the opposite of Cluster 1, which started to rise gradually from March to stabilize. 8,394 (29.6%) active developer contribution trends fall into this cluster. This pattern suggests that there are some active developers who have become more active after the outbreak. For example, one developer with the account name ‘pavangoal42’ contributed mostly under 10 per day in the first three months of 2020, and increased significantly from April to July, with an average daily contribution of 17.7.

Cluster 3 (in blue) implies a relatively stable trend with little change in the contribution before and after the outbreak. This seems to capture the normal state of work for developers, and they are hardly affected by COVID-19. Roughly a half of the active developers (48%) are in this cluster. For example, a developer with the account name ‘lambert-p’ has been contributing in a balanced manner in 2020, in a typical weekly work pattern, with more contributions during weekdays and less or no contributions on days off.

V. DISCUSSION

A. Implications

Our investigation addresses the need to provide scholarly evidence concerning how the COVID-19 pandemic affected GitHub developers’ contributing activities. On the whole, we register a significant increase in developer activity and contributions on GitHub in the early stages of the COVID-19 global explosion. On an individual country basis, the trend
in GitHub developer contributions is closely related to the outbreak situation in the corresponding country. Our findings also show that working from home as practiced by companies does not affect the commitment of software developers to GitHub. Besides, developers with different levels of activeness have made more contributions during COVID-19, especially inactive developers. To conclude, our research implies that GitHub, as a typical representative of open source communities and sharing platforms, plays an important role in the face of public crisis. It should be noticed that these technologies and resources can be very helpful in other emergencies too. Relevant participants (software engineering practitioners and researchers, etc) should take note of these findings and understand the strengths and usefulness of GitHub in COVID-19 pandemic, in order to make fuller use of it as a powerful weapon in our response to the crisis.

B. Limitations and Future Work

We recognize that our study carries several limitations and potential threats to validity. First, some of our conclusions rely on case studies, such as country-level and organization-level analyses, and do not include the full range of data. This is mainly because our preliminary study did not concern a lot of workload and made it a focus for future work. Second, this study aims to analyze the impact of COVID-19 on the open source community, however GitHub is not the only platform where people can share their open source projects, which might limit our observations. Third, we only consider the coarse-grained contribution of each developer. Although the overall contribution is representative enough to reflect the activeness of developers, enabling fine-grained analysis of each kind of contribution can offer us more insights, which will be studied in the future. Forth, we acknowledge that there may be some non-developer users in our dataset since GitHub is being used for purposes other than software development. However, it is impractical for us to identify whether each user is a real developer or not. Nevertheless, all the developers considered in this paper have created at least one public repository (see Section III), thus we believe that the observations in this paper could reflect the general behaviors of GitHub developers.

VI. CONCLUSION

This research focuses on GitHub developers’ contributions and presents the first large scale empirical study of the impact of COVID-19 on the activity and contributions of GitHub developers. We go to great lengths to collect a dataset of over 25 million GitHub developers and characterize them from four perspectives including the overall ecosystem level, country level, organization level and developer level, to understand the impact of COVID-19 on the open source software community. Our observations suggest that COVID-19 does not present a challenge for GitHub developers, and show the promising direction of applying open source technologies and resources to response to public emergencies.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (grant numbers 62072046 and 61702045). Haoyu Wang is the Corresponding author.

REFERENCES

Analyzing Program Comprehensibility of Go Projects

Moumita Asad, Rafed Muhammad Yasir, Shihab Shahriar, Nadia Nahar, Md. Nurul Ahad Tawhid
Institute of Information Technology, University of Dhaka, Bangladesh
{bsse0731, bsse0733, bsse0703, nadia, tawhid}@iit.du.ac.bd

Abstract—Program comprehension is one of the most important activities in developing and maintaining software. Although existing studies have examined aspects of Go such as design patterns, code smells and comment density, the comprehensibility of Go has not been explored yet. This study analyzes the comprehensibility of Go by comparing it with Java based on five metrics namely Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments. For comparison, 50 popular, diverse and open-source projects are selected from each language. Results show that Go projects outperform Java in terms of Nesting Depth, Lack of Cohesive Comments and Duplicate Comments.

Index Terms—Go, Java, Program comprehensibility, Software maintenance.

I. INTRODUCTION

Go is an open-source programming language developed and maintained by Google [1]. At present, the popularity of Go is rising rapidly and many teams are adopting it for their projects. Renowned projects such as Docker and Kubernetes have been written in Go. Existing studies [1], [2], [3] have analyzed aspects of Go such as design patterns, code smells and comment density. However, the comprehensibility (the degree of ease with which a programmer read and comprehend a program [4]) of Go projects have not been explored yet.

Program comprehension is the precondition of performing any maintenance related activities [5]. It consumes more than half of the maintenance resources [6]. A major reason behind introducing the Go programming language is to make code more maintainable [7]. To make Go projects maintainable, code must be comprehensible [4]. Therefore, analyzing the program comprehensibility of Go projects will help to gain insight regarding their maintainability and identify scopes for making the projects more comprehensible. Furthermore, it will help developers to decide whether they should use Go for their projects [6].

In this context, the current study aims at investigating the comprehensibility of projects written in Go. Additionally, a tool named CompreGo has been developed to detect code fragments that need to be refactored for improving comprehensibility. To analyze the comprehensibility of Go, it is compared with Java. The reason is that both of these languages are object-oriented languages [2] and both of these belong to the C-family (e.g., C++, C#) [8]. Furthermore, an existing study concluded that Java projects have higher program comprehensibility compared to C, C++ and C# projects [9]. This study measures program comprehensibility using five static code metrics namely Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments (non-informative comments) and Duplicate Comments [9], [10]. These metrics are language-independent and found to be a good indicator of program comprehensibility. Based on these metrics, Wilcoxon Rank-Sum Test [11] is used to analyze comprehensibility of Go and Java projects.

For analysis, 50 popular, diverse and open-source projects are selected from each language based on several criteria such as projects cannot be a fork, and it must have a description or readme file [9]. Results show that Go projects have significantly better comprehensibility than Java in terms of Nesting Depth, Lack of Cohesive Comments and Duplicate Comments. However, Java projects have significantly fewer long methods than Go. Regarding Too Long Files, no significant difference is found between the two languages.

II. METHODOLOGY

This study analyzes the program comprehensibility of Go projects by comparing those with Java projects. The study is conducted in 4 steps namely dataset selection, dataset preprocessing, comprehensibility metrics calculation and statistical analysis. At first, Go and Java projects are selected based on several criteria such as availability of readme files. Next, relevant files (e.g., production code) are selected from the projects to achieve better comparability of results [9]. After that, comprehensibility metrics of the code residing in these files are calculated. Based on these metrics, statistical analysis is performed to identify if the comprehensibility of Go and Java projects are significantly different. The details of these steps are given below:

1) Dataset Selection: This study selects Go and Java projects that fulfill the following criteria:
   - To filter out dummy projects, repositories having at least 10,000 lines of code and a readme file or a GitHub description are selected, as followed in [9].
   - To avoid duplicate projects, repositories must not be a fork or mirror of another one.
   - An existing study found that almost 75% projects above 215 MB contain the same code multiple times [9]. Besides, 99% repositories in the GHTorrent dataset (a collection of GitHub repositories) [12]...
are below this threshold. Therefore, the size of the codebase should not exceed this threshold.

2) Dataset Preprocessing: After selecting the projects, folders containing external libraries, generated source files or test code are excluded to solely focus on the core code of the repository. Although test codes are important, those are excluded from analysis to improve comparability between the repositories because it is found that developers may not apply high quality standards to it [9]. To filter out these files, a list of 60 file system paths (e.g., "**/libs/**", "**/test/**", "**/testsuite/**") provided in [9] is used. The entities of the list do not depend on the organization or naming convention followed by the Go or Java repositories. Apart from the list, the pkg folder is excluded from Go projects since it contains internal libraries that are not part of the main source code.

3) Metrics Calculation: To measure program comprehensibility, five static code metrics namely Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments are used. These metrics are language-independent and found to be a good indicator of program comprehensibility [9]. These metrics are calculated by inspecting source code files and parsing the corresponding Abstract Syntax Tree.

- **Too Long Files (TLF):** Long files are difficult to comprehend since a large fragment of code needs to be reviewed [9]. These files can indicate bad modularization as well. Too Long Files measures the portion of source code lines that are located in files exceeding 750 lines [9]. For calculating this metric, (1) is used.

$$ TLF = \frac{\sum LOC \ of \ long \ files}{total\ LOC} \times 100 \ (1) $$

Where, total LOC denotes LOC of the whole project.

- **Too Long Methods (TLM):** Similar to long files, long methods are difficult to comprehend since a large code fragment needs to be examined [6], [9]. Too Long Methods calculates the number of code lines that reside in methods surpassing 75 lines [9]. It is measured using (2).

$$ TLM = \frac{\sum LOC \ of \ long \ methods}{total\ LOC} \times 100 \ (2) $$

- **Nesting Depth (ND):** Deeply nested codes are hard to comprehend as the interleaving control structure of the code needs to be considered, as shown in Listing 1 [13]. Nesting Depth measures the fraction of code lines belonging to methods exceeding nesting depth 5 [9]. It is calculated using (3).

$$ ND = \frac{\sum LOC \ of \ deeply \ nested \ methods}{total\ LOC} \times 100 \ (3) $$

- **Lack of Cohesive Comments (LCC):** Comments lack cohesiveness when they are non-informative [10]. A sample comment is shown in Listing 2, where the comment conveys obvious information. Lack of cohesive comments can confuse and mislead developers and increase the cost to comprehend and maintain code [14].

```java
if(marks<40) {
  result = "fail";
} else {
  if(marks>=80) {
    result = "A";
  } else if(marks>=60) {
    result = "B";
  } else {
    result = "C";
  }
}
```

Listing 1. Nesting Depth

```java
/**
 * @return true or false
 */
protected boolean isLoginTicketBased() throws Exception {
  ..........
  ..........
}
```

Listing 2. Non-cohesive Comment

To compute Lack of Cohesive Comments, the name and lead comment of each method are tokenized using camel case format and white-space respectively [14]. Next, the similarity between these two token sets is measured using Levenshtein distance [15]. Two tokens are considered similar if the distance is smaller than 2 [10]. To calculate coherence, the total number of similar words are divided by the total number of comment words. If the value of coherence is 0 or above 0.5, the comment is non-cohesive [10]. Lastly, Lack of Cohesive Comments is calculated using (4).

$$ LCC = \frac{total \ non-cohesive \ comments}{total \ comments} \times 100 \ (4) $$

Where, total comments represents the total number of method lead comments in the project.

- **Duplicate Comments (DC):** Existing studies show that source code often contains methods with duplicate comments [14], [16]. These comments do not provide meaningful information on the different implementation of methods and thereby increases the difficulty in program comprehension [14]. Calculation of Duplicate Comments follows a similar process as Lack of Cohesive Comments. Duplicate
comments are searched within a file since they are more likely to occur in the same file [16]. At first, all the comments of a file are tokenized and stored in lists. For each comment pair, Levenshtein distance [15] is calculated between elements of their corresponding token lists. Two comments are considered duplicate if the distances of all the elements are less than 2 [10]. Lastly, Duplicate Comments are measured using (5).

\[ DC = \frac{\text{total duplicate comments}}{\text{total comments}} \times 100 \] (5)

All of the above metrics are measured in percentage and a higher value indicates lower comprehensibility.

4) **Statistical Analysis:** After calculating the metrics, Wilcoxon Rank-Sum Test [11] is conducted to check whether the comprehensibility of Go and Java projects are significantly different. Wilcoxon Rank-Sum Test is used because it makes no assumption regarding the distribution of the data (e.g., normal distribution) and it can handle unequal sample size (e.g., different number of duplicate comments for Go and Java projects) [11]. The null hypothesis is “Code written in Go and Java have no differences regarding comprehensibility”. This hypothesis is tested individually for each of the five comprehensibility metrics discussed above.

III. **Supporting Tool: CompreGo**

To calculate comprehensibility metrics for Go, a tool named CompreGo is developed which is publicly available 1. Developers can use the tool to measure the comprehensibility of a project and identify which fragments need to be refactored for improving comprehensibility.

- **Comprehensibility Metrics of a Project:** To find comprehensibility metrics (Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments) of a project, run the following command, where `directory` is the project path.

  ```
  ./comprego -d directory
  ```

  The output will show metrics value in percentage, as displayed in Fig. 1. Here, Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments are 15.92%, 10.69%, 4.55%, 17.45% and 6.06% respectively.

  ![Fig. 1. Comprehensibility Metrics of a Project](image)

- **Custom Thresholds for Metrics Calculation:** The default thresholds for Too Long Files, Too Long Methods and Nesting Depth are 750, 75 and 5 respectively [9]. However, each project and its maintainers are different and thus thresholds may need to be changed. For this reason, CompreGo has a provision for customizing these thresholds. These thresholds can be changed by adding the following arguments while running the tool:
  - “-lf number”: Set customized threshold for Too Long Files
  - “-lm number”: Set customized threshold for Too Long Methods
  - “-nd number”: Set customized threshold for Nesting Depth

  Where, `number` represents the customized threshold value.

IV. **Experimentation and Result Analysis**

For comparing the comprehensibility of Go and Java, 50 popular, open-source projects of each language are selected from GitHub that fulfill the criteria presented in Section 2 [17]. The popularity of a project is measured based on the number of stars, as followed in [18], [19]. Table I shows the top 5 projects from each language along with their number of stars and LOCs. For example, project Guava has 40000 stars and its LOC is 756954.

![Fig. 2. List of Long Methods](image)

To compare the projects, Wilcoxon Rank-Sum Test [11] is conducted separately for each of the five comprehensibility metrics (Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments), discussed in Section 2. The results are presented in Table 2. The mean of Too Long Files is slightly higher in Go, however, there is no significant difference between the two languages (p-value = 0.39 > 0.05). In terms of Too Long Methods, Go has longer methods than Java. This is because
of the error handling mechanism followed by these two languages. In Java, errors are mostly dealt with try catch blocks. On the contrary, errors in Go are handled using if statements. Whenever a function is called in Go, it usually returns an error in its return values [8]. The errors returned by the callee function are handled with an if block inside the caller function. For example, the function PreparedQueryResolve() in Listing 3 returns ErrMissingQueryID as an error and the caller function Explain() in Listing 4 handles the error using an if statement. Although Java can handle multiple errors using one catch block, Go associates individual if statement for each function call. Consequently, more error handling statements are written inside Go method than Java which increases the method size. Through analysis, it is found that on average each Java method contains 0.05 error handling statements if statement. Although Java can handle multiple errors using one catch block, Go associates individual if statement for each function call. Consequently, more error handling statements are written inside Go method than Java which increases the method size. Through analysis, it is found that on average each Java method contains 0.05 error handling statements (catch block), whereas it is 0.88 in Go. Besides, Go does not support generics for increasing simplicity and making code more readable. However, this also makes the code more verbose compared to other languages and results in longer files and methods [2].

### TABLE I
**DESCRIPTIO**N OF TOP 5 JAVA AND GO PROJECTS

<table>
<thead>
<tr>
<th>Language</th>
<th>Project</th>
<th>Star</th>
<th>LOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>Elasticsearch</td>
<td>53200</td>
<td>2566030</td>
</tr>
<tr>
<td></td>
<td>Spring Boot</td>
<td>53000</td>
<td>363943</td>
</tr>
<tr>
<td></td>
<td>RxJava</td>
<td>44100</td>
<td>467397</td>
</tr>
<tr>
<td></td>
<td>Guava</td>
<td>40000</td>
<td>75954</td>
</tr>
<tr>
<td></td>
<td>Retrofit</td>
<td>37400</td>
<td>36610</td>
</tr>
<tr>
<td>Go</td>
<td>Moby</td>
<td>59400</td>
<td>1396039</td>
</tr>
<tr>
<td></td>
<td>Hugo</td>
<td>49400</td>
<td>144222</td>
</tr>
<tr>
<td></td>
<td>Gin</td>
<td>48800</td>
<td>19968</td>
</tr>
<tr>
<td></td>
<td>Fio</td>
<td>42300</td>
<td>20771</td>
</tr>
<tr>
<td></td>
<td>Gogs</td>
<td>36200</td>
<td>90034</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Metric</th>
<th>Mean (Go)</th>
<th>Mean (Java)</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Too Long Files</td>
<td>25.61</td>
<td>18.99</td>
<td>0.39</td>
</tr>
<tr>
<td>Too Long Method</td>
<td>17.59</td>
<td>5.69</td>
<td>0.00</td>
</tr>
<tr>
<td>Nesting Depth</td>
<td>5.78</td>
<td>4.49</td>
<td>0.00</td>
</tr>
<tr>
<td>Lack of Cohesive Comments</td>
<td>18.47</td>
<td>23.69</td>
<td>0.00</td>
</tr>
<tr>
<td>Duplicate Comments</td>
<td>2.89</td>
<td>5.43</td>
<td>0.00</td>
</tr>
</tbody>
</table>

1 The significance level is 0.05

In terms of Nesting Depth, Go code have fewer nesting than Java. The reason again lies in the way how errors are handled in these languages. In Java, try blocks are used to handle code that may throw an error. When a try block is added in a method, it increases the nesting depth, as shown in Listing 5. Results reveal that around 33.14% deeply nested method in Java contains at least one try catch block. To find the association between deeply nested method and the presence of try catch block, Chi-square Test [20] is conducted since both of these are categorical variables. The result shows a significant association between deeply nested method and the presence of try catch block (p-value < 0.05). However, there is no concept of try catch blocks in Go [8]. Rather, errors are handled by adding if statement inside the caller function. The if statement checks the error and return values accordingly, as presented in Listing 4. Since the if statement exist in the same nesting depth as it’s neighbouring code, it does not increase the nesting depth of the function. As a result, functions in Go are less nested than Java.

```go
func (s *Store) PreparedQueryResolve() (uint64, *structs.PreparedQuery, error) {
    if queryIDOrName == "" {
        return 0, nil, ErrMissingQueryID
    }
}
```

Listing 3. Returning error in Go

```
func (p *PreparedQuery) Explain() error {
    _, query, err :=
        state.PreparedQueryResolve()
    if err != nil {
        return err
    }
}
```

Listing 4. Handling error using if statement in Go

```java
@Override public void run() {
    while (!__isClosed) {
        try {
            if ((ch = __read(true)) < 0) {
                break;
            }
        } catch (InterruptedException e) {
            synchronized (__queue) {
                __ioException = e;
                __queue.notifyAll();
            }
        }
    }
}
```

Listing 5. Handing error using try catch blocks in Java

Go shows better results in Lack of Cohesive Comments and Duplicate Comments as well. This can be due to the difference in documentation generation process from the source code comments [21]. Java and Go use different comment structures and separate tools to generate documentation. In Java, documentation is generated through the Javadoc tool [21]. It is necessary for comments in Java to be annotated with Javadoc tags (e.g., @param, @return, @author) to produce meaningful documentation. In Go, documentation is generated through the Godoc tool [8]. Unlike Javadoc, Godoc does not require tags in comments for generating meaningful documentation. It only needs a comment preceding a code and its description in plain words. Godoc then uses the comment and as much information it can get from the code to generate documentation. This makes the process of documenting Go code much easier than Java. Additionally, the community considers that the process of generating documentation from comments is simpler and
easier in Go than in other languages\(^2\). Such advantages may motivate developers to write better comments that are more cohesive and not duplicates of other comments.

V. Threats to Validity

This section presents potential aspects that may threaten the validity of the study:

- **Threats to external validity**: The analysis is conducted on 50 Go and 50 Java projects which are selected using several criteria such as availability of readme file or having at least 10,000 lines of code. Although the project selection criteria are based on an existing study [9], the obtained results may not generalize to other projects. However, to mitigate the threat of generalizability, diverse and popular projects are selected, as followed in [17], [22].

- **Threats to internal validity**: Threats to internal validity include errors in the implementation and experimentation. The first threat to internal validity lies in measuring program comprehension. Although research on program comprehension started more than 30 years ago, till now there is no well-defined metric to measure it [23]. This study uses five static code metrics namely Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments to measure program comprehensibility. An existing study found that these metrics are language-independent and a good indicator of program comprehensibility [9]. However, changing the metrics (e.g., psychological factors, dynamic metrics [23]) may impact the obtained results. The second threat to internal validity comes from setting the thresholds of the comprehensibility metrics. This study set the threshold for Too Long Files, Too Long Methods and Nesting Depth to 750, 75 and 5 respectively. These thresholds may vary depending on the context. However, these thresholds are adopted by a previous study [9] as well.

VI. Related Work

Existing work related to this paper can be broadly classified into two categories - studies related to program comprehension and studies related to Go, which are discussed in the following subsections.

A. Studies Related to Program Comprehension

Although research on program comprehension started more than 30 years ago, there are still no well-defined metrics to measure it [23]. However, various researches [5], [9], [24] have been conducted in this regard. Scalabrino et al. used 121 metrics to investigate their correlation with program comprehension [5]. These metrics are related to code (e.g., cyclomatic complexity), documentation (e.g., methods internal documentation quality) and developer experience (e.g., years of programming experience in any language). However, the study found that none of the metrics show a significant correlation with program comprehension.

Roehm et al. studied 10 conventional wisdom related to software maintainability more specifically program comprehension [9]. For example, C code has more too long methods than code written in other languages. They used 5 metrics - Clone Coverage, Comment Incompleteness, Too Long Files, Too Long Methods and Nesting Depth for measuring program comprehension. However, another study countered that code clones are helpful for program comprehension [25]. In addition, writing comments are not enough if they convey unrelated or inconsistent information [14]. Therefore, this paper uses Lack of Cohesive Comment and Duplicate Comments instead of Comment Incompleteness. Furthermore, unlike this study, their analysis does not consider Go programming language.

B. Studies Related to Go

Prior studies related to Go can be divided into two categories. The first category [3], [22] includes Go projects as a part of their dataset. Ray et al. compared 17 programming languages including Go to find whether the choice of language affects software quality [22]. Their analysis confirms that language has a small yet statistically significant impact on code quality. Besides, classifying bugs into several categories (e.g., Memory or Concurrency error), the paper examined whether language influences the type of bug that occurs in software. The result revealed that defect types are strongly associated with languages. For example, languages with managed memory systems (e.g., Java) naturally had fewer memory errors or leaks compared to unmanaged languages (e.g., C).

Another study conducted by He et al. explored differences in commenting practices across different languages [3]. They analyzed the comment density of 5 popular programming languages namely Python, Java, Go, JavaScript and C++. Their study revealed that Python and Java projects have significantly higher comment density than C++, JavaScript and Go projects. In addition, the purpose of a project (e.g., reuse, application, education) impacts its comment density. For example, educational projects have the highest comment density.

The second category [1], [2] conducts research on Go from various perspectives. Schmager et al. analyzed the design patterns of Go [2]. They implemented all the 23 Gangs of Four (GoF) design patterns and compared these with Java. They found that Go’s language features have not replaced design patterns. Implementing the adapter pattern is easier in Go. On the other hand, implementing the template pattern is difficult since there is no abstract class in Go. Furthermore, although Go syntax is an improvement over C++ or Java, it is more verbose than Python or Haskell.

\(2\)https://blog.golang.org/godoc-documenting-go-code

259
Another study by Yasir et al. proposed a tool named God-Expo to detect God Structures (a structure that threatens the maintainability and understandability of code by performing most of the work alone) in Go programs [1]. GodExpo uses three metrics namely Weighted Method Count (WMC), Tight Class Cohesion (TCC) and Access To Foreign Data (ATFD) for detecting God Structures. Besides, it can provide version wise result to observe the evolution of God Structures. By executing GodExpo on Go projects, the authors showed that it can detect God Structs in all versions of a project. Additionally, it was found that number of God Structures in a project gradually increases as a project evolves.

The above discussion indicates that various studies have been conducted on Go. However, none of these studies focuses on the maintainability or comprehensibility of Go. Therefore, this paper aims at analyzing the comprehensibility of Go. In addition, it proposes a tool CompreGo to help developers in tracking the comprehensibility of Go projects.

VII. Conclusion and Future Work

This paper examines the comprehensibility of Go projects by comparing it with Java projects. For measuring program comprehensibility, five static code metrics namely Too Long Files, Too Long Methods, Nesting Depth, Lack of Cohesive Comments and Duplicate Comments are used. Based on these metrics, 50 popular, diverse, open-source Go and Java projects are compared using Wilcoxon Rank-Sum Test. Results demonstrate that Go code has significantly higher comprehensibility than Java in terms of Nesting Depth, Lack of Cohesive Comments and Duplicate Comments. Conversely, Java has significantly better comprehensibility than Go in terms of Too Long Methods. Regarding Too Long Files, no significant difference is observed between these two languages. In future, the comprehensibility of Go projects will be analyzed from other perspectives such as dynamic analysis or psychological aspect. In addition, refactoring suggestion will be developed for improving comprehensibility of Go projects.

REFERENCES

What Do Developers Reply To? An Empirical Study of the Top Unmanned Aerial Vehicles (UAVs) Apps

Fatma Outay  
College of Technological Innovation (CTI)  
Zayed University  
Dubai, United Arab Emirates  
Fatama.outay@zu.ac.ae

Haroon Malik, David Dampier,  
College of Engineering and Computer Sciences (CECS)  
Marshall University  
Huntington, WV, USA  
Malikh, dampierd@marshall.edu

Abstract—Most modern UAVs/drones have a dependency on using a mobile device as part of the flight systems. The drone manufacturers launch the controlling app for the drones in mobile app stores. There has been a tremendous upsurge in the number of UAVs (Unmanned Aerial Vehicle) mobile apps on the Google Play Store over the past few years. A UAV/Drone app user expresses the experience with the app by providing comments on the Google Play Store. Whereas, the developers interact with the user by posting replies to users' comments. Feedback in users' comments enables developers of mobile apps to discover issues, such as features requests, bugs to fix, and app maintenance requests, in a timely manner. The value of responding to a user review of drone apps has never been explored. We conducted the largest and most extensive empirical study of UAV mobile apps to-date by analyzing the 1,825 UAV mobile apps across twenty-five categories, with 162,250 reviews (user comments and developers' replies). We categorized the developer replies into seven major categories. We also find that 35% of the developer's replies are associated with providing direct solutions to the drone app user's problems. Whereas, only 1% of the developer replies are related to the app’s monetary refund issues.

Keywords-component; Developer reply, Mobile App, Google Play Store, Drones, UAV

I. INTRODUCTION

Google Play, referred to as the Android Market, is Google’s official store and portal for Android apps, games and other media content for the Android OS phone, tablet or Android TV device. Purchases made on the Google Play store can not only be shared and synced across mobile devices but can also be downloaded and stored on the Google cloud. As of 2017, Google Play features over 3.5 million Android applications, with 2.8 million apps available for download presently.

Within the past five years, i.e., recently, there has been a tremendous increase in the number of Unmanned Aerial Vehicles (UAVs) (also known as a drone; a term used interchangeably with UAV in the paper) apps on the google play store. The UAV is an aircraft without a human pilot on board. UAVs can be navigated via control from the ground through software-controlled flight plans in their embedded systems, functioning along with onboard sensors and GPS. Small UAVs mostly use lithium-polymer batteries, while larger vehicles rely on conventional airplane engines. Some of these drones are equipped with cameras that allow the user to record videos or capture pictures. These drones are controlled by certified operators. Also, there exist many drone hobbyists.

A. Importance of Developer Reply

After interaction with drone apps, users can instantly express their experience with an app and can directly communicate with developers via app reviews; a direct channel between users and developers. Feedback in users’ comments enables developers of mobile apps to discover issues, such as features requests, bugs to fix, and app maintenance requests, in a timely manner [6][18]. Perceiving the importance of mobile apps user feedback and developers interaction with the uses via comments, Google Google Play Store, and Apple App store have established a review response system that:

(a). Allows the developers to respond to an app user review quickly. The developer in the response can address the user concern, provide details of the app functionalities, acknowledge users’ feature requests, or simply thank the user for their feedback and

(b). Quickly notify the user who posted the review, the developer’s reply/feedback, and provide an option to update the corresponding app review [2][7].

It has been established via empirical studies [9][15][16] that developers provide feedback/response to mobile app users in a timely and accurate manner to:

(a). Improves user experience and

(b). Increases the app popularity and rating

B. Problem Statement

The pace of innovation in the drone industry is increasing at a tremendous rate. Thousands of companies compete globally, with more emerging every day. Besides the explosion in production, prices have reduced exponentially. Additionally, drones are being incorporated with an array of sensors, cameras, and software applications. Yet, there exists no study to-date highlighting user issues related to drone apps that can assist app store stakeholders, especially in producing quality drone apps. As well, examine the developer replies to these complaints to better understand (a) the areas/user concerns that receive enough attention and (b) the areas/user concerns that are important to users but not well attended by the developers. There have been a few studies in the literature aiming at software review-related research, they all neglect the developer's replies in their analysis.
Thus, the paper (a) analyzes the developer’s responses (as comments) to drone app users, (b) categorizes the developer’s answers into types based on the user issues being handled, and (c) systematically examines and provides details of the category.

II. RELATED WORK

Previous works on reviews of mobile apps affirm that user reviews play a vital role in the triumph of an app. Kim et al. [12] found that reviews of an app influenced its purchase the most. Ha and Wager [8] found that users who are evaluating the apps are doing so only when they are either extremely satisfied or dissatisfied. Fu et al. [5] have performed Topic Modelling on 1 and 2-star ratings to discover the most common types of complaints in each category of apps. Khalid et al. [11] studied app store 1 and 2-star reviews to identify what iOS app users are doing so only when they are either extremely satisfied or dissatisfied. Khalid et al. [11] studied low-rated reviews of 20 free iOS apps and found that Functional Error, Feature Request, and App Crash were the categories that were reviewed most often. In contrast, privacy and ethics, feature removal and hidden costs complaints were the most impactful ones. Martin et al. [14] provided a survey paper that contains a more exhaustive list of studies conducted on iOS Apps. Hassan et al. [9] studied 4.5 million reviews with 126,686 developer responses of 2,328 top free apps from the Google Play Store. The study was an attempt to (i) explore more about the dynamic nature of the review-response mechanism and (ii) find if responding to a review often has a positive effect on the rating that is given by the user. Noei et al. [17] studied 435,628 reviews from 49 apps (across 10 categories) from the Google Play Store, performing topic modeling to identify the categories of user feedback. In contrast, Mahmoudi et al. [13] studied only the review of 19 Android wearable apps and concluded that Functional Errors and Cost categories receive the most complaints. In contrast, Installation Errors, Device Compatibility, and Privacy & Ethical Issues are the ones causing a higher negative impact on app ratings. Hu et al. [10] investigated 68 hybrid apps from the Google Play Store and iOS app store to determine whether they achieve consistent star ratings and user reviews across app platforms.

III. DATA COLLECTION

We searched the entire Google Play Store for all the mobile apps with the terms ‘Drone’, ‘UAV’, ‘Drone Controllers’, ‘Drone Simulators’ and ‘Drone Games’. We executed a Breadth-First-Search and crawled all the related apps, including their ‘docid’, ‘hreflink’, ‘developer’, ‘app price’, ‘app summary’ and ‘app score’ using an open-source scraper [4]. To get more comprehensive coverage and a large number of drone apps for the empirical study, the free and paid apps were scraped separately using the price: ‘free’ and price: ‘paid’ options in the search method provided by the scraper. A total of 1825 drone apps were collected.

IV. ANALYSIS OF DEVELOPER REPLY

The percentage of developer replies according to their categories is depicted in Figure 1. Below, we analyze each category and provide details of the issues discussed and responded by the drone app developers.

A. Provide Solution

In this category, the drone developer provides a number of steps for the app user to follow and solve the issue. For example:

“In case of connectivity problems, please make sure no other DJI apps are running in the background. To do so, either restart your phone or go to the settings app/applications and terminate them (closing them with home button is not enough).”

Often these types of developers' responses are accompanied by direct support mail contact information or forum links, asking the customer to revert if the steps do not solve the issue. For example,

“Please try the below steps and revert at support@reliancegames.com if the issue persists

1. Kill/close the game and any unnecessary apps running in the background
2. Free up some space in the device/Clear game cache
3. Reboot/restart your device and launch the game in a strong network (WIFI)”

Many drone app users are concerned with the extent of private information the drone app wants to access, and they feel unsafe about it. Thus, most of the developers' responses in this, i.e., ‘provide solution’ category, clarify the user, why the drone app needs to collect some of the personal information [3]. Many developers in this category have justified their in-app purchases, informing the users that the in-app purchases were mentioned in the “app description” and directed them to read it.

B. Request Detail

The ‘request detail’ category makes up 15% of the developer’s replies (included in the study). This category includes developer’s replies in which they seek more information from the drone app users about the encountered problem. Such as, provide details about their setup, mobile device model, drone

![Figure 1. Percentage of developer replies based on the perceived categories](image1.png)

![Figure 2. A sample developer response to a UAV mobile app user.](image2.png)
model (e.g., DJI drone model), firmware version (e.g., DJI firmware version) and app version. In order to troubleshoot the drone app users problems, some developers ask functionality queries like: is the user able to launch the app, go past the start-up screen, or is the app freezing while doing so? E.g., “Hello, please contact support@flylitchi.com so we can help you. Make sure to include details about your setup (mobile device model, DJI drone model, DJI firmware version, Litchi version).”

A few developers in this category asked the drone app users whether the issues reported happened after an app update. In a few replies, developers requested the drone app users to send all the app’s crash logs. Some developers also asked the drone app users to post screenshots or send videos of the issue, e.g., “Thank you for advising of the experience you encountered Stefanos. Were currently investigating into this occurrence with your device specifications. But if you would be able to provide any screenshots of any error message or disconnected status in the application, please email support@airmap.com!”

The developers have also asked some users to report their issues to the community forum page, customer care email id, or to their technical support department call facility.

For the app users who have requested a refund for the app, developers have asked them to provide more details, such as their request order number, to initiate the refund process. Similarly, many reviews that mentioned that the drone app needs more improvement, the developers requested the users to: (a) Provide more detail on what improvements are to be made and (b) Provide/send a concrete suggestion to the support team.

4.3. A Solution in Progress Notification

The comments in this category are the drone app’s developer’s replies to inform the app users that their engineers are investigating the reported issue(s), and/or a solution is in progress. Around 9% of developer replies belong to this category. Most of the developer’s responses in this category state that: (a) The next update will fix the issue reported in the user feedback. (b) They will consider supporting additional drone models for their app either in the next app update or release. (c) They will add compatibility for other/additional devices and android OS versions in a future update.

As a response to new feature addition requests, the developers acknowledged the customers for their valuable feedback. They stated that the development team is actively working on the feature addition or would consider their recommendations as they continuously strive to improve their drone app. As for bug fixes, most developers apologized for the delay and requested users to kindly extend their patience and follow until the issue was resolved. In case of the excessive advertisement complaints, the developers responded that the concerned team is being notified to resolve the issue, e.g.,

“Thank you for your feedback. The question is under investigation.”

“Hello, we do apologize for any inconvenience. We already reported the compatibility and crash issue to the development team, and they are working right now on the application to resolve the issue as soon as possible.

Kindly extend your patience. Best Regards, Parrot Community Support.”

C. Offer Refund

A very meager percentage of only 1% of the developer responses offered a full refund to the users who are dissatisfied with the drone app. In some reviews, users have complained that they did not receive a refund. For all such complaints, the developers have asked the users to provide their order number or transaction id since they would not initiate a refund without it. In contrast, a few developers asked for the purchase receipt to be sent to the support via email. Most of the developers who offered a refund (in their replies) were associated with drone education apps where the discontented user reports that the practice questions provided in the app were not helpful and resulted in the user failing the drone pilot exams. In the case of device incompatibility issues, the developer in their response apologized to the customer, offering a refund and stating that it is impossible to provide support for all the varieties of drones or devices present in the market, e.g., “One star? If it’s that bad send me an email. More than happy to issue a refund.”

“Please contact us at support@appologics.com and we can help you with issues or re-fund.”

D. Offer Direct Support

In response to negative user feedback, most of the developers offered direct support (35%) i.e., asked users to contact them directly (through the contact details provided instead of reporting app issues in the Google Play Store platform) so that they can guide them on how to use certain features of the app. In cases, where the developers could not provide direct support (due to company policies), all of them apologized to the user for the displeased experiences they incurred with the drone app. Further, developers guided the users through possible options to receive support, such as listing the steps to report their issues to the technical support team via support mail id (e.g., support@flylitchi.com), or phone number of a support forum community (e.g., https://www.facebook.com/spacewargame) or relevant website (e.g., https://www.parrot.com/support/hotline) where the user can report their issue with details to the support executive and receive immediate solutions to their issues, e.g.,

“Please contact support@flylitchi.com so we can help you fix this.”

“Hello, users, you can consult. jov@simtoo.com Thank you.”

E. Solved Notification

Solved Notification Category includes all the developer responses/comments aimed to notify the drone apps users that the issue reported in their review (negative feedback) has been solved. Among all the developer replies, only 9% of replies belong to this category, which implies that drone developers do not solve customers’ issues actively, or they take too long to respond, or they do not bother to notify the user even after the issue is solved. Among all the developers who reply in this category, 82% of replies were to notify that their reported problem was solved as part of a new release of software and the user needs to upgrade to the new version, e.g.,
“Thanks for your review. There was a bug. I just uploaded an update (version 2.4.1), which fixes it again.”

Only 6% of the developer’s replies in the category include a notification to the user that their issues have been explicitly taken care of. In all such responses, the developer advised the users to log out of the drone app and log back in for new changes to come into effect.

“We appreciate your patience during this fix Chris. I am happy to report that the Air-Map application is currently functioning properly and displaying airspace information when selecting a particular gridded area for recreational and commercial operators. Fly Safe!”

Some developer responses in this category, i.e., 4%, advise users that the issue reported is tied to the old version of Android OS, not the mobile drone app. Updating to the new/latest Android OS update will resolve their issues. The remaining 9% of the developer replies in this category include: (a) Notifying drone app users of added support for new drone models or new phone compatibilities as requested by them (b) Apologizing to the drone app users for the service outage and notifying that the service has rebounded and (c) Expressing regret for the delay in responses since the app improvisation was a major update.

F. Others

Almost 12% of the developer responses could not be classified into any of the above five categories due to their arbitrary nature of responses. For example, many of the developer replies were not in regard to addressing any of the user’s top-27 complaints, as shown in Figure 4, rather simply thanking and appreciating the users for their feedback. For most of the good reviews in 3-star rating comments, which did not specifically mention any complaints but just appreciated the app, the drone developer replies to express their gratitude and explicitly asked the happy and satisfied users to consider giving a harsh response. However, analysis of the developer's replies yields that 6% of the developer's replies in this category are counter-replies, apologizing for the inconvenience caused by the drone apps (such as by the Parrot SA developers).

6. CONCLUSION AND FUTURE WORK

This paper is a large-scale empirical study of UAV or drone-related apps of the Google Play Store Platform. The study consisted of 1,825 UAV mobile apps, across twenty-five categories, with 162,250 reviews. We find that most top drone apps do not respond to reviews. However, responding can lead to a positive change in rating. Addressing specific issues and notifying the users that requested features are available are most likely to lead to a change in the review rating. As future work, we plan to undertake the following tasks: (a) identifying the stakeholders of the reviews and perform review analysis on the individual stakeholder perspective, (b) perform similar review analysis on UAV or drone-related apps of other mobile app platforms such as the iOS store, Blackberry World, and (c) Apply machine learning on the manually categorized user reviews to enable the automatic classification of upcoming new user feedback into their respective complaint types.

Funding: This research was made possible by NASA WV EPSCoR, Grant # 80NSSC20M0055.

REFERENCES

Quantifying Synergy between Software Projects using README Files Only

Roxanne El Baff  
Institute for Software Technology  
German Aerospace Center (DLR)  
Oberpfaffenhofen, Germany  
roxanne.elbaff@dlr.de

Sivasurya Santhanam  
Institute for Software Technology  
German Aerospace Center (DLR)  
Cologne, Germany  
sivasurya.santhanam@dlr.de

Tobias Hecking  
Institute for Software Technology  
German Aerospace Center (DLR)  
Cologne, Germany  
tobias.hecking@dlr.de

Abstract—Software version control platforms, such as GitHub, host millions of open-source software projects. Due to their diversity, these projects are an appealing realm for discovering software trends. In our work, we seek to quantify synergy between software projects by connecting them via their similar as well as different software features. Our approach is based on the Literature-Based-Discovery (LBD), originally developed to uncover implicit knowledge in scientific literature databases by linking them through transitive connections. We tested our approach by conducting experiments on 13,264 GitHub (open-source) Python projects. Evaluation, based on human ratings of a subset of 90 project pairs, shows that our developed models are capable of identifying potential synergy between software projects by solely relying on their short descriptions (i.e. readme files).

Index Terms—repository mining, natural language processing

I. INTRODUCTION

The growing amount of open-source software projects range from small experimental software to large-scale and continuously advancing systems. Many of such projects are available on public repository hosting platforms such as GitHub that do not only provide functionalities for managing source code but also include tools for documentation and collaboration.

Thus, repository hosting platforms are not only technical means but can also be considered as an agglomeration of ideas and knowledge scattered over several projects. However, exploring this vast amount of information manually is beyond human capacity.

Several applications assisting users in exploring software repositories aim at finding similar repositories to one’s own repository, e.g. [1–3], helping to identify alternative implementations, explore related projects, or identify plagiarism. However, when focusing solely on similarities, searchers are trapped in their search bubbles and are rarely exposed to develop something new based on repositories that complement their work. Thus, in this paper, we propose an approach for exploiting distributed software knowledge in repository platforms, which is inspired by Literature-Based Discovery (LBD) [4]. LBD uncovers implicit knowledge and synthesizes hypotheses from scientific literature databases by identifying complementary information sources (i.e., publications). The general idea is to seek possible transitive relationships between concepts mentioned in different publications. For example, if concepts A (e.g., a disease) and B (e.g., an enzyme) are found to be often related to a concept C (e.g., a drug) respectively, but no relationship between A and B has been reported so far, one can hypothesize that there is also a relationship between A and B which is worth to be explored.

In this paper, we adopt the idea of LBD to quantify a synergy score between project pairs taking into account, both, software feature similarities and differences. For example, Figure 1 illustrates a simplified scenario of two repositories with a potential synergy. Repository A has two features: (1) “pre-process data for significance tests”, and (2) “conduct significance tests”. Whereas, Repository B has only feature (2) as common one and an additional feature: (3) “conduct post-hoc analysis”. As we see, these two repositories can benefit from each other because they share a common feature, and each is missing one feature from the other. Accordingly, we ask the question: How can one discover software repositories that are similar to some extent but different enough to expand each other’s functionality so that they can be the basis for new developments?

Our approach solely relies on information extracted from repositories’ publicly available readme file because it is a common practice, in high-quality open-source projects, to summarize the main project information in such files. More precisely, we use existing natural language techniques to model the software features of each project, then we quantify...
the synergy between each project pair by defining a new synergy ranking method.

We conduct our experiments on more than 13K GitHub software repositories with Python as the main programming language and having a readme file written in English. We evaluate our models based on human ratings, which shows that our approach successfully identifies synergy between project pairs.

Altogether, our contribution is threefold:

- A new view on exploring synergy quantification between repositories to inspire new ideas.
- A novel approach that combines existing natural language techniques to extract relevant information from repositories’ descriptions, and ranking techniques to quantify synergy between pairs of projects.
- Experimental evidence that synergy between software projects can be detected automatically.

Our work can be used, among others, for recommendation systems, or discovery-based systems from large software projects pile. For reproducibility, the code is publicly available: https://github.com/DLR-SC/repository-synergy.

II. RELATED WORK

A. Recommendation of software repositories

Previous works in the repository mining community focus already on developing approaches for software recommendations by either using the metadata of GitHub repositories (e.g., stargazes, readme files, …) such as Zhang et al. [3], or by using software artifacts (e.g., software packages, code, …) such as McMillan et al. [1]. Others focus on categorizing software repositories and readme files, which helps to better perceive a massive pile of data and grasp the content faster (Prana et al. [5], Sharma and Thung et al. [2], …).

Zhang et al. [3] built a recommendation system called RepoPal. They detect similar repositories using three different heuristics based on readme files and stargazing. They assume that two repositories are likely to be similar based on three measurements: 1) readme files with similar content, 2) repositories starred by users of similar interests, and 3) repositories starred together within a short period by the same user. Their recommendation system outperform CLAN (Closely reLated ApplicatioNs) [1]. McMillan et al. [1] developed an approach for automatically detecting similar applications for a given Java application based on packages and class hierarchy.

Our goal and methodology differ from Zhang et al. [3]. In our approach, we exploit not only software similarities but also differences which we believe can inspire for new directions. Moreover, our methodology is different: we exploit the implicit knowledge between software projects based only on readme files instead of relying on different metadata.

B. Cataloging software repositories

Another strand of research in repository mining essential to our work is categorizing software repositories’ thematic analysis of readme files. GitHub creates showcases where they manually catalog a set of repositories on a certain topic. Sharma et al. [2] semi-automatically expanded such showcases. Using 10K repositories with readme files, they first extract the most descriptive section in the readme file by selecting the one with the highest cosine similarity value with the repository short description on the top of the repository landing page on GitHub. They then feed all these descriptions to a Latent Dirichlet Allocation using the Genetic Algorithm model, where they manually analyze topics into meaningful categories. This work indicates that readme files are already used in existing research to deduce the software features of a software repository. Also, using a topic modeling algorithm is common for clustering readme files and hence software repositories. In this work, we use similar techniques within our approach. However, we identify sections reflecting the software features in readme files using an existing classifier, READMEClassifier, trained by Prana et al. [5]. [5] systematically classify each section of a readme file to categories reflecting its purpose (see Section III-A for more details).

III. APPROACH

The following section outlines our approach to quantify synergies between software repositories, which consists of a pipeline with three steps, as shown in Figure 2.

A. Software Features Extraction

The first step extracts the repositories’ descriptions of software features (Figure 2.1). To do that, we identify these sections by using the existing multi-label classifier, READMEClassifier, built by Prana et al. [5], which labels readme file sections. READMEClassifier was trained on 4k manually annotated readme file sections from 393 repositories with an F1 score of 0.75 where each section was categorized into one or more of eight different categories. The What category is identified based on headings (e.g., About) or based on the text at the beginning of a README file. Sections describing a comparison to another software artifact, with respect to performance, flexibility, and simplicity are categorized as Why. Other categories describe other metadata not related to software features. Prana et al. [5] combined What and Why into one label (Why sections were rare < 3%), WhatWhy, to train their READMEClassifier.

We use this classifier to extract sections with the label WhatWhy for each repository, which describes the software features within it. In the next steps, we only use repositories having a readme file with WhatWhy section(s).

B. Software Features Modelling

Topic modeling is an unsupervised machine learning technique that automatically analyzes text data to determine cluster terms for a set of documents (in our case, readme files). Each document is assigned a weighted sum of topics. And each topic is represented by a set of terms and the probability of this term for a specific topic.
1. Inverse Feature Overlaps

\[ M = \exp(-X^T \times X) \]

\[
\begin{array}{cccc}
F1 & F2 & F3 & F4 \\
F1 & 0.6 & 0.8 & 1 & 1 \\
F2 & 0.8 & 0.6 & 0.8 & 1 \\
F3 & 1 & 0.8 & 0.3 & 0.8 \\
F4 & 1 & 1 & 0.8 & 0.2 \\
\end{array}
\]

2. Repository-Feature Graph

\[
X = R1 \times F1 \\
R2 \times F2 \\
R3 \times F3 \\
R4 \times F4 \\
R5 \times F5 \\
\]

3. Repository Similarities

\[
O = X \times X^T \\
\begin{array}{cccc}
R1 & R2 & R3 & R4 \\
R1 & 0.6 & 0.2 & 0 & 0 \\
R2 & 0.2 & 0.4 & 0.1 & 0 \\
R3 & 0 & 0.4 & 1 & 0.3 \\
R4 & 0 & 0.1 & 0.3 & 0.7 \\
R5 & 0 & 0 & 0 & 0.7 \\
\end{array}
\]

C. Synergy Quantification

Now that we have a vector representation for each repository, we describe here the approach for finding software project pairs with synergies using these vectors.

As mentioned previously, we base our synergy scoring on the ABC model of Literature-Based-Discovery. In the domain of software projects, A, B, and C represent software features. We first formalize the problem, and we define three requirements for our synergy scoring approach. We then suggest a random walk-based ranking function for repository pairs, which comply with these requirements.

1) Problem formalisation: From the previous step, each repository a is represented by a vector \( x^{(a)} \in \mathbb{R}^{|F|} \) with elements \( x_{a,i} \) giving the association strength of a to feature \( i \). All vectors are assembled as the rows of the repository-feature matrix \( X \in \mathbb{R}^{|R| \times |F|} \). An example is shown in Figure 3.2.

2) Synergy Scoring Requirements: Synergy is a subjective notion; therefore, we suggest three main requirements for quantifying synergy between software project pairs:

- **R1 - Potential trend.** Two software projects should each bring features that were not combined by many other projects before (create a new trend). More formally, high values \( m_{i,j} \) in the inverse feature overlap matrix \( M = \exp(-X^T X) \) (Figure 3.1) has high values for feature pairs that are not frequently present in the same projects.

- **R2 - Potential complementary features.** For two software projects \( a \) and \( b \) to bear potential to create new directions when combined, it is required that one has strong associations to a subset of features for which the other has weak associations. This means for two complementary repositories \( a \) and \( b \) \( \exists_{i,j \in F} : x_{a,i} \uparrow \land x_{b,j} \downarrow \land x_{a,j} \downarrow \land x_{b,j} \uparrow \). The dissimilarity, and thus the potential to have complementary features, of repositories is summarized in the matrix \( X^{-1} = 1 - X^T X \)

\( X \) is always row normalized.

For the sake of simplicity we denote \( \exp(-X^T X) \), the element-wise application of the exponential function to the negative values of the matrix \( X^T X \)
• R3 - Similarities between projects. R1 and R2 alone are not sufficient to discover the synergy between software project pairs because they can lead to matches between projects from very different domains and purposes. Thus, two projects that benefit from each other should also have some common characteristics. This can be expressed by the row-normalised repository-feature overlap matrix $O \in \mathbb{R}^{[R] \times [R]}$ (Figure 3).

3) Synergy Scoring as Ranking functions: We define a synergy ranking function for pairs of software repositories that comply with the requirements defined above based on a restricted random walk on a heterogeneous graph of repositories and features similar to the ones shown in Figure 3 when the matrices are treated as transition probability matrices.

The transition probability matrices $P(X) \sim X$ and $P(X^T) \sim X^T$ assign probabilities to go from a repository to a feature or vice versa, respectively. The probability of moving from a feature node in the graph (Figure 31) to another feature node is given by $P(M) \sim M$.

The idea of random walk-based ranking is that a random walker starts at a repository $a$ and randomly jumps to an associated feature $i$ with probability $p(x_{a,i})$. In the next step, it jumps, with probability $1 - d$, to another repository that is also affiliated with feature $i$, which accounts for commonalities between the two. With probability $d$, it jumps (or explores) another feature not well related to $a$ and $j$ by moving according to the probability $p(x_{a,j})p(x_{a,j})$. From there, it discovers another repository $b$ with high affiliation to feature $j$ (according to the probability $p(x_{b,j})$). The resulting matrix equation for synergy scores of all repository pairs is:

$$Q_{rw} = (1 - d)P(X)P(X^T) + d(P(X)P(M) \circ X^T)P(X^T)$$

The jumping probability $d$ can be adjusted to balance between finding similar repositories and exploring new features.

IV. EXPERIMENTS

In this section, we first describe the dataset used in our experiments (Section IV-A) and then we describe our experiments.

A. Data

For our experiments, we exploit GitHub open-source repositories. We use GitHub repositories because of the availability of the data and tools to extract their metadata and readme files. We limit the extracted repositories to one main programming language, Python, to control the variability of programming language-software features dependability, which is outside the scope of our work. We use the latest dump from the GHTorrent dataset (Gousios, 2013 [6]). To ensure high repository quality, we rely on the number of “Watchers” (> 50) for each repository. “Watchers” are GitHub users who have asked to be notified of activity in a repository. So, the high number of watchers reflect repositories with high quality or ones relevant to the community. Also, to ensure recency, we fetch repositories that are still available on GitHub, and were updated in the recent year. Lastly, we fetch the readme files using PyGithub[3] where we end up with 20,590 repositories’ readme files.

B. Software Features Extraction

As mentioned in Section III-A, we classify 20,590 readme files containing 169,521 sections using the READMEClassifier. Table 1 shows the total number of repositories and sections after each pre-processing step. Based on the READMEClassifier, only 14,065 have at least one WhatWhy section. We, then, filter out the non-English sections by using the langdetect Python library [7]. We end up with 13,264 readme files (24,988 WhatWhy sections), which are used in the subsequent steps.

C. Software Features Modelling

In this section, we transform the 13,264 readme files, that include only what-why sections, into a numeric vector by using topic modeling techniques, as described in Section III-B.

We use Mallet latent Dirichlet allocation (LDA) [8]. We choose the optimal $k$ (pre-set number of topics) configuration based on the coherence value, which assesses the quality of the learned topics by measuring the degree of semantic similarity between high-scoring terms in a topic.

LDA is a generative probabilistic model for a collection of discrete data such as text corpora (here, readme files). The model defines a set of topics to describe a corpus. Each document is modeled as a finite mixture over an underlying set of topics that are represented as a mixture of terms. Here, the association vectors of a what-why section of a repository’s readme file to topics build the repository-feature matrix $X$ used in the next phase.

First, we pre-process the readme files by removing the stop-words and lemmatizing the content. After that, we train Mallet LDA on the readme files (WhatWhy sections) where the number of topics ($k$) must be predefined. So, we train models with $k$ ranging between 40 and 150 topics. To define the optimal $k$, for each value, we calculate the average of topic coherence [10] values of the inferred topics. Figure 4 plots the...
average of the coherence values for different $k$. As we see, the topics are most coherent where $k = 45$. Also, Table II gives a sample of the topics’ terms.

D. Synergy Quantification

Using the modeled features from LDA, we apply here the ranking algorithm, random walk, with different configurations of the jumping probability $d$, $d = [0.0 - 0.5]$. Using each configuration, we calculate the score of each repository-pair, sorted in descending order, in our dataset. We evaluate these models in the next Section V.

V. Evaluation

A first observation is that random walk-based ranking with a larger jumping probability ($d > 0.2$) results in a similar set of repositories $r_p$ that are rated to have high synergy with any other repository $r_q$. Therefore, we only evaluate 1) a model with random-walk jumping probability $d = 0.0$ (focusing on similarity only), 2) with jumping probability $d = 0.2$ to see the effect of exploring similar yet more different features, and 3) random selection of repository pairs (baseline). We hypothesize that our models (1 and 2) identify synergy between repository pairs better than the random pair selection.

Due to the absence of ground truth data and comparable methods, we conducted a study where we asked programmers with good proficiency in English (the language of readme files) and high knowledge of Git and Python$^3$ to rate repository pairs regarding their potential for bearing synergy.$^4$ We created a dataset of 90 repository pairs containing 30 non-overlapping top picks of the three models mentioned above ($d = 0.0$, $d = 0.2$, random selection). The 90 pairs were divided into three different batches, where each batch contained ten pairs of each model. Each batch was rated by three different raters. As a result, we obtained 270 evaluations.

As shown in Table III we asked our raters to evaluate each repository pair by reading their readme files and then answering two questions. In question 1, we ask if there is a synergy between the pair. The possible answers ranged from None to Strong. Also, the raters briefly explained the rationale of their choices (question 2). The formulation of questions and the rating guidelines were refined in a pilot study prior to the main evaluation.

Table V shows that the majority agreement (2 out of 3) between our human raters is very high when synergy intensity is considered. However, the full agreement between all 3 raters is often not achieved, which indicates that there is a degree of subjectivity in human judgment.

VI. Results

Table VI shows the count of the synergy evaluations for repository pairs selected by the different models. We observe that repository pairs picked randomly have only 23% (21 annotations) of annotations indicating synergy, whereas the repository pairs generated by our models have higher synergy reports of 66% and 58% for models 1 and 2, respectively. The difference to the baseline is significant with regard to the non-parametric Kruskal-Wallis Test $^1$ (not normally distributed data), $p < 0.001$. This was further confirmed in a post-hoc analysis using the Mann-Whitney test $^2$ with Bonferroni correction that showed that the repository pairs generated by model 1 and model 2 are significantly rated with higher synergy than the random pairs at $p < 0.001$ and effect sizes, $r$, of 0.30 and 0.39 respectively. However, the difference between models 1 and 2 is not significant. Table IV shows examples of repository pairs rated high by our models.

While overall, model 1 ($d = 0.0$), focusing on similarities of project pairs, have a higher agreement with human ratings, a closer look at the highest rated pairs by humans from the entire evaluation set shows a different picture. We define the discovery rate, $dr(p_{top})$, as the intersection of repository pairs belonging to the top $p$ ($p \in [0,1]$) fraction of repository pairs ranked by human annotators ($top_{human}$) and the algorithm ($top_{algo}$) relative to the number of top pairs, that is: $dr = \frac{|top_{human} \cap top_{algo}|}{n}$, where $n$ is the number of all pairs in the evaluation dataset. For model 1, $dr(p_{top} < 2) = 0.0$. While for model 2 ($d = 0.2$), $dr(p_{top} = 0.1) = 0.1$ and $dr(p_{top} = 0.15) = 0.08$. This indicates that model 2 incorporates differences of features more strongly. Contrarily, model 1 scores higher synergy for pairs that have more redundancies, which makes them more obvious to the raters.

VII. Conclusion

This paper explored a novel approach for discovering synergies between software projects that may inspire innovations. To this end, we adapted the idea of Literature-Based Discovery (LBD), which aims at uncovering implicit knowledge by

---

$^1$Raters studied for their bachelor/masters degree in English and have $> 3$ years experience in Python.

The aim is to build a discovery system that assists in identifying every new knowledge combination that leads to a useful finding. The same applies to systems that use our approach. If the discovery for novelty identification, it is acceptable that not every new knowledge combination leads to a useful finding.

Exploring similarities and differences of knowledge artifacts, to the software domain. Based on human rating evaluation for identifying synergy between pairs of software projects showed that it is possible to quantify synergy using projects’ readme files only. Our results indicate that models focusing on similarities to identify synergy are slightly higher rated by humans.

However, in the original spirit of Literature-Based Discovery for novelty identification, it is acceptable that not every new knowledge combination leads to a useful finding. The same applies to systems that use our approach. If the aim is to build a discovery system that assists in identifying new directions for novel developments, one would put more emphasis on differences in software features. The developed methodology is, however, flexible enough to be configured to identify repository pairs that serve similar purposes.

### TABLE III

**The questions that our raters had to answer after reading the readme files of two repositories.**

<table>
<thead>
<tr>
<th># Questions</th>
<th>Answers</th>
</tr>
</thead>
</table>
| 1. I see that there is synergy between the 2 repositories | a. None – No complementary or common features  
| | b. Weak – More common features than complementary  
| | c. Somewhat – Some features can be merged  
| | d. Strong – Clear complementary features that lead to a new project |
| 2. Explain your choice(s) (Keep it short) | Free text |

### TABLE IV

**Examples of Repository pairs majorly annotated as having strong synergy** (clear complementary features that lead to a new project), ranked by LDA-RW for \(d = 0.0\) and \(d = 0.2\).

<table>
<thead>
<tr>
<th>Jumping Probability</th>
<th>Repository 1</th>
<th>Repository 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d = 0.00)</td>
<td>Flexible and scalable Django authorization backend for unified per object permission management</td>
<td>Core common behaviors for Django models, e.g. Timestamps, Publishing, Authoring, Editing and more. Blender python addon to increase workflow for creating minecraft renders and animations</td>
</tr>
<tr>
<td>(d = 0.2)</td>
<td>Python scripts and documentation for generating topographically accurate Minecraft maps from historical map scans</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE V

**Majority and full agreement between human scoring for repository pairs selected by different models.**

<table>
<thead>
<tr>
<th>Synergy Intensity</th>
<th>Synergy vs. Majority</th>
<th>No Synergy Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA - Random Walk ((d=0.0))</td>
<td>80%</td>
<td>20%</td>
</tr>
<tr>
<td>LDA - Random Walk ((d=0.2))</td>
<td>83%</td>
<td>20%</td>
</tr>
<tr>
<td>random baseline</td>
<td>100%</td>
<td>50%</td>
</tr>
</tbody>
</table>

### TABLE VI

**Counts of the rated synergies for the 90 repository pairs in our dataset, for each algorithm (LDA Random walk with \(d = 0.0\), with \(d = 0.2\) and random baseline). Each pair was rated by three raters.**

<table>
<thead>
<tr>
<th>Synergy w Has Synergy</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDA - Random Walk ((d=0.0))</td>
<td>31</td>
<td>12</td>
<td>25</td>
<td>22</td>
<td>31</td>
<td>59</td>
</tr>
<tr>
<td>LDA - Random Walk ((d=0.2))</td>
<td>38</td>
<td>15</td>
<td>25</td>
<td>12</td>
<td>38</td>
<td>52</td>
</tr>
<tr>
<td>Baseline random</td>
<td>69</td>
<td>06</td>
<td>11</td>
<td>04</td>
<td>69</td>
<td>21</td>
</tr>
</tbody>
</table>

### References


Evaluating a Tool for Creating Bug Report Assignment Recommenders

Disha Devaiya, John Anvik, Meher Bheree, Farjana Yesmin Omee
Department of Mathematics and Computer Science
University of Lethbridge, Alberta, CANADA
E-mail: devaiya86@gmail.com, [john.anvik, bheree, omee]@uleth.ca

Abstract

Large software development projects that use bug tracking systems can become overwhelmed by the number of reports filed. To assist in reducing the workload of project members, researchers have proposed the use of bug report assignment recommenders. To assist project members with the creation of assignment recommenders, we proposed a web-based tool called the Creation Assistant for Supporting Triage Recommenders (CASTR). This paper presents the results of both a laboratory and field study of CASTR. We found that CASTR can create assignment recommenders with accuracy as high as 95%, 80%, and 70% for Top-1, Top-3 and Top-5, respectively. The field study showed that 60% of the participants found CASTR easy to use, whereas the remaining participants found CASTR moderately or slightly easy to use.

1. Introduction

An issue tracking system plays an important role in the development of a high-quality software product. Such systems record information for a bug report or feature request. These records include the name of the developer that resolved the issue and other relevant development activity. Issue tracking systems are particularly important when team members are globally distributed [3].

During the testing phase of software development, a developer or tester confirms that the software is working per the specifications. If irregularities are found, a triager marks it as a bug in the issue tracking system and includes such information as the steps to reproduce and screenshots. A triager will then assign the reported bugs to the appropriate developer based on the view of the developer’s ability or their bug fixing history. In the case of large projects, a large number of new bugs can be submitted daily [2]. For a software project that uses a manual triage process, the bug triagers can become overwhelmed. To address such problems, researchers have proposed the use of bug report assignment recommenders [3, 4, 6, 7, 8].

However, the creation of a bug report assignment recommender for a software project can be a complex and time-consuming process. To address this problem, we previously proposed the Creation Assistant for Easy Assignment (CASEA) [1]. CASEA was further refined into a web-based tool called the Creation Assistant for Supporting Triage Recommenders (CASTR) [5]. CASTR allows a project member to create an assignment recommender for a project by specifying such items as labelling heuristics, a machine learning algorithm and a data imbalance technique. They can also use a “specify and verify” approach to determine the optimal configuration settings for a project-specific assignment recommender. By providing a little project knowledge, a project member can produce a bug report assignment recommender in a short period.

This paper presents an evaluation of the CASTR system to answer the following research questions:

1. RQ1: Does CASTR create assignment recommenders that make accurate recommendations? If CASTR creates assignment recommenders that make accurate recommendations, then a triager need not examine the report as deeply. Using such recommenders changes the triager’s role from making decisions relying on their knowledge, experience, intuition or information they can gain from existing tools to confirming decisions made by the recommender. This shift changes, and hopefully, reduces triager cognitive load.

2. RQ2: Can human triagers make effective use of information presented by CASTR? If CASTR creates assignment recommenders that assist human triagers then time can be saved by not assigning bug reports to the appropriate developer manually. Some of the human resources consumed by the triage process can be then directed elsewhere in the project.
Our analytical evaluation conducted using bug report datasets from the Plasmashell\(^1\), LibreOffice\(^2\) and Firefox\(^3\) projects showed that recommenders with good accuracy could be created. A field study with ten participants from different technical backgrounds gave evidence that they were able to make effective use of the information provided and that most of them are likely to use CASTR in creating an assignment recommender.

2. Creation Assistant for Supporting Triage Recommenders (CASTR)

CASTR \(^5\) is a platform-independent web-based tool that assists a project member with the creation of bug report assignment recommenders. It provides a web interface for downloading a dataset from a Bugzilla repository. Information about the collected dataset is displayed by a Configuration tab. CASTR assists with setting project-specific heuristics for labelling reports with the names of the developer to be recommended. As not all of the developer names may be valid, CASTR allows the user to select a minimum threshold of resolution activity to eliminate developers that have resolved a small number of bug reports. CASTR provides the option of choosing one of four supervised machine learning algorithms: Support Vector Machines (SVM), Multinomial Naïve Bayes, C4.5, and Rules.

We chose these algorithms as they represent different categories of supervised machine learning algorithms. CASTR also provides three different approaches to handle imbalanced data: oversampling using SMOTE, manual oversampling and undersampling using Expectation-Maximization (EM). When the user clicks the “Recommender” button in the Configuration tab, a request is sent to the CASTR web service and redirects the user to an Analysis tab that presents progress information, such as the time to train the recommender and evaluation results. The Analysis tab displays the average Top-1, Top-3 and Top-5 precision and recall values for a testing set. A user can tune a recommender by comparing these values with the last five generated recommendations.

3. Evaluation

In evaluating CASTR, we sought to answer our two research questions. We answered RQ1 using an analytical evaluation of the recommenders created by CASTR and investigated RQ2 through a field study.

3.1. An Analytic Evaluation of the Recommenders

We selected bug reports with the resolution Fixed from three open source projects: Plasmashell, LibreOffice and Firefox. Also, we removed bug reports for developers who fixed less than 20 reports. We used a 90%/10% split between training and testing sets. Table 1 shows the number of bug reports used for the three projects. The third column shows the number of bug reports remaining filtering.

We evaluate the performance of a recommender using the metrics of precision\(^4\) and recall\(^5\). This requires us to know the set of developers who could have been accurately assigned to a bug report. We approximated this information using the names of developers who fixed reports in the same component as the testing bug report.

We explored all possible combinations of machine learning and data imbalance algorithms provided by CASTR. As Table 2 shows, we found that LibreOffice produced the highest precision values (97/95/91) with the Naïve Bayes/SMOTE combination whereas for Firefox the Naïve Bayes/undersampling with EM technique was the best (55/36/36). On the other end, Plasmashell had the best results (96/83/73) using an SVM algorithm without any imbalance technique applied. The low recall values are likely a result of overestimating the set of possible developers. For example, in the Firefox project, it was not uncommon for the estimated set of developers to be 30+ developers, meaning that the best recall value for a single recommendation would be about 3% (1/30).

3.2. A Field Study of CASTR

To answer our second research question, we conducted a field study with experienced software developers, project managers, bug triagers and graduate students. The study contained ten (10) participants: 3 project managers, 5 application developers and 2 graduate students.

The field study was conducted by first asking participants to complete an initial survey that collected demographic information and technical background details. The demographic questions were for general analysis to break down the response data into meaningful groups. For example, we found that Indian participants took the survey more than the participants with other nationalities and the majority were in the age range of 26 to 39. Most participants completed a graduate-level of schooling, and most of them belong to the job function Application Developer. For technical background, 60% reported having a lot of prior experience with issue tracking systems, and 33% had a good amount of experience with contributing to large open-source projects.

---

\(^1\)https://bugs.kde.org
\(^2\)https://bugs.documentfoundation.org
\(^3\)https://bugzilla.mozilla.org

---

\(^4\)Precision measures how often the approach makes a relevant recommendation for a report

\(^5\)Recall measures how many of the relevant recommendations are truly recommended
Table 1. Training and testing set sizes for evaluating recommenders.

<table>
<thead>
<tr>
<th>Project</th>
<th>Original Dataset Size</th>
<th>After Filtering Dataset Size</th>
<th># of Bug Reports in Training Set</th>
<th># of Bug Reports in Testing Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plasmashell</td>
<td>1112</td>
<td>532</td>
<td>479</td>
<td>53</td>
</tr>
<tr>
<td>LibreOffice</td>
<td>2500</td>
<td>1725</td>
<td>1553</td>
<td>172</td>
</tr>
<tr>
<td>Firefox</td>
<td>1000</td>
<td>777</td>
<td>699</td>
<td>78</td>
</tr>
</tbody>
</table>

Table 2. Evaluation results of assignment recommender

<table>
<thead>
<tr>
<th>Project</th>
<th>Algorithm</th>
<th>Sampling Technique</th>
<th>Precision Top 1</th>
<th>Precision Top 3</th>
<th>Precision Top 5</th>
<th>Recall Top 1</th>
<th>Recall Top 3</th>
<th>Recall Top 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plasmashell</td>
<td>SVM</td>
<td>None</td>
<td>96</td>
<td>83</td>
<td>73</td>
<td>11</td>
<td>26</td>
<td>38</td>
</tr>
<tr>
<td>LibreOffice</td>
<td>Naïve Bayes</td>
<td>SMOTE</td>
<td>97</td>
<td>95</td>
<td>91</td>
<td>2</td>
<td>7</td>
<td>11</td>
</tr>
<tr>
<td>Firefox</td>
<td>Naïve Bayes</td>
<td>UnderSampling/EM</td>
<td>55</td>
<td>36</td>
<td>36</td>
<td>11</td>
<td>15</td>
<td>9</td>
</tr>
</tbody>
</table>

Overall, participants had low bug triaging experience\(^6\) and low familiarity with the machine learning algorithms.

After completing the initial survey, participants were given a user manual of CASTR and asked to create an assignment recommender for the Plasmashell project. The Plasmashell dataset used was identical to that used for the analytical experiment.

In the field study, a total of 71 recommenders were created by the participants using different heuristic configurations provided by CASTR. Table 3 shows the quantitative results for the best recommender created by each participant. The first column identifies the unique participant. The second column presents the machine learning algorithm selected by the participants for creating their best assignment recommender. The next two columns present the minimum and maximum threshold that the participants selected before creating their most accurate assignment recommender using CASTR. Half of the participants chose values greater than or equal to 10 for the minimum threshold and the other half used values less than 10. By default, the maximum threshold is set to the largest activity value depending on the set project-specific heuristics. These values were left unchanged by the participants. The next three columns show the Top-1, Top-3 and Top-5 precision and recall values for the best assignment recommender created by each participant. Most of the assignment recommenders were created using the SVM machine learning algorithm. In the case of Top-5, two scenarios have no values as the recommender suggested less than 5 developers because of threshold value settings.

As a part of the recommender evaluation, CASTR provides information about how long the tool takes to create a recommender. For most users, the average time to create a recommender using any of the algorithms was less than 30 seconds. In general, the results show that assignment recommenders created using the C4.5 and SVM algorithms took more processing time than the assignment recommender created using the Naïve Bayes and Rules algorithms. Two notable exceptions were for Users 1 and 7, whose average recommender creation times using C4.5 was between 160 and 200 seconds. A possible reason for the large processing time is that both participants had set the minimum threshold value as 1, meaning that CASTR considered all of the possible developers for classification, which led to a substantial increase in the processing time.

Although participants were provided with a video presentation of CASTR and a brief tutorial of the recommender creation process at the beginning of the field study, participants encountered some problems related to understanding concepts, specifically with labelling bug reports and setting an appropriate minimum threshold value. Most of the participants did not initially understand how to select the appropriate label for bug report resolution and which machine learning algorithm to use. Also, the meaning of the precision and recall metrics was not initially well understood by participants. However, once their meaning was understood, participants felt they made more intelligent choices.

The field study results show that 60% of the participants found CASTR easy to use whereas the remaining participants found CASTR moderately or slightly easy to use. We received positive responses about recommending CASTR for creating a recommender for bug report assignment with 50% of the participants responding “very likely” or “extremely likely”, and the remaining participants responding “moderately likely”. When asked whether they believed that the assignment recommenders created using CASTR would reduce the time to triage bug reports, all participants responded either “extremely likely” (2), “very likely” (5), or “moderately likely” (3).

Based on observations while analyzing the field study result, participants were found to employ two strategies for as-

---

\(^6\)A possible reason for this is that most of the participants were part of a large software development project team where their responsibilities are limited to within specific modules or feature development.
4. Conclusions

This paper presented the results of our evaluation of CASTR, a tool to assist software development projects with the creation of bug report assignment recommenders. Our analytical evaluation showed that CASTR can create recommenders with good accuracy, answering RQ1.

Our field study demonstrated that users were able to create accurate bug report assignment recommenders in less than 10 trials. This indicates that they were able to make good use of the information provided by CASTR, answering RQ2. Also, users found the tool was generally easy to use.

Although the results we obtained have shown that a CASTR assists the project members with the creation of assignment recommenders based on feedback and the results of the user study, several future improvements were identified. These include extending CASTR to collaborate with the other issue tracking systems, having the tool use information from duplicate bug reports, supporting the creation of other types of triage recommenders, and a field study by dataset project members.

Table 3. Best assignment recommenders created by participants.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Algorithm</th>
<th>Trials to Best</th>
<th>Threshold Min</th>
<th>Threshold Max</th>
<th>Top 1 (%) Precision</th>
<th>Top 1 (%) Recall</th>
<th>Top 3 (%) Precision</th>
<th>Top 3 (%) Recall</th>
<th>Top 5 (%) Precision</th>
<th>Top 5 (%) Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>User1</td>
<td>SVM</td>
<td>2</td>
<td>5</td>
<td>108</td>
<td>92</td>
<td>9</td>
<td>83</td>
<td>25</td>
<td>72</td>
<td>34</td>
</tr>
<tr>
<td>User2</td>
<td>SVM</td>
<td>1</td>
<td>10</td>
<td>130</td>
<td>96</td>
<td>11</td>
<td>83</td>
<td>25</td>
<td>77</td>
<td>38</td>
</tr>
<tr>
<td>User3</td>
<td>SVM</td>
<td>1</td>
<td>26</td>
<td>112</td>
<td>94</td>
<td>11</td>
<td>85</td>
<td>30</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>User4</td>
<td>Naïve Bayes</td>
<td>5</td>
<td>49</td>
<td>73</td>
<td>87</td>
<td>9</td>
<td>78</td>
<td>21</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>User5</td>
<td>SVM</td>
<td>1</td>
<td>10</td>
<td>130</td>
<td>96</td>
<td>11</td>
<td>85</td>
<td>25</td>
<td>77</td>
<td>38</td>
</tr>
<tr>
<td>User6</td>
<td>SVM</td>
<td>2</td>
<td>1</td>
<td>141</td>
<td>74</td>
<td>17</td>
<td>58</td>
<td>33</td>
<td>56</td>
<td>49</td>
</tr>
<tr>
<td>User7</td>
<td>SVM</td>
<td>4</td>
<td>1</td>
<td>61</td>
<td>91</td>
<td>7</td>
<td>85</td>
<td>21</td>
<td>74</td>
<td>29</td>
</tr>
<tr>
<td>User8</td>
<td>Naïve Bayes</td>
<td>1</td>
<td>1</td>
<td>117</td>
<td>50</td>
<td>5</td>
<td>37</td>
<td>10</td>
<td>33</td>
<td>14</td>
</tr>
<tr>
<td>User9</td>
<td>SVM</td>
<td>1</td>
<td>10</td>
<td>115</td>
<td>94</td>
<td>10</td>
<td>82</td>
<td>24</td>
<td>76</td>
<td>38</td>
</tr>
<tr>
<td>User10</td>
<td>SVM</td>
<td>7</td>
<td>1</td>
<td>102</td>
<td>87</td>
<td>14</td>
<td>65</td>
<td>27</td>
<td>60</td>
<td>42</td>
</tr>
</tbody>
</table>

References

A family of experiments for evaluating the usability of a collaborative modelling chatbot

Ranci Ren  
Dep. Ing. Informática  
Univ. Autónoma de Madrid  
Madrid, Spain  
ranci.ren@estudiante.uam.es

John W. Castro*  
Dep. Ing. Informática y Ciencias de la Comunicación  
Universidad de Atacama  
Copiapó, Chile  
john.castro@uda.cl

Silvia T. Acuña  
Dep. Ing. Informática  
Univ. Autónoma de Madrid  
Madrid, Spain  
silvia.acunna@uam.es

Abstract—Recent natural language processing developments have facilitated the adoption of chatbots in typically collaborative software engineering tasks. Families of experiments can overcome limitations in terms of the sample size of individual experiments. To experimentally evaluate the usability of a chatbot for collaborative modelling (i.e., SOCIO) and tackle some of the typical shortcomings of individual experiments, we conducted a family of three experiments to evaluate the usability of SOCIO against the Creately online collaborative tool. Results show that the participants were more satisfied with the chatbot than with the online collaborative tool and that they also created class diagrams faster using the chatbot. We conclude that chatbots may be helpful for creating class diagrams.

Keywords—Chatbots, Family of Experiments, Usability, Modelling

I. INTRODUCTION

Modelling is a fundamental part of the software development process, and it is often a collaborative activity [1]. A plethora of cloud-based platforms have recently emerged for synchronous mechanisms (e.g., Lucidchart, Gliffy and Creately). The SOCIO chatbot, a collaborative modelling tool, was developed to provide an alternative method for building models or meta-models using Twitter or Telegram (nick @ModellingBot) [2]. Along with the SOCIO chatbot, users benefit from social network collaboration and ubiquity to perform the lightweight modelling task [2].

Experiments can assess the effectiveness of software engineering (SE) treatments (e.g., tools) and check whether or not the hypotheses about the effectiveness of such treatments hold. Unfortunately, isolated experimental results may be unreliable due to small sample sizes [3], while families of experiments increase the reliability of joint conclusions and rule out the detrimental effects of publication bias on conclusions [4][5]. It is critical to assess chatbot usability because they are increasingly being used across many domains [6][7], and poor interactions would have an impact on user willingness to use the service [8]. To increase the reliability and generalizability of individual experimental results, we used a family of experiments to assess the usability of the SOCIO chatbot.

In our family of three experiments, we compared the usability of the chatbot SOCIO with Creately (https://creately.com/app). Creately is a real-time collaborative tool built on Adobe’s Flex/Flash technologies. We chose Creately as the control tool since no previous studies had assessed Creately usability, even though it is the most used online collaborative modelling tool [9], and it has similar functionality to SOCIO. Along the way, we made several findings with respect to efficiency, effectiveness and satisfaction issues in response to our research question:

RQ: Compared to Creately, does the use of SOCIO positively affect user efficiency, effectiveness, and satisfaction with respect to class diagram construction in a family of experiments?

Our findings contribute: (1) empirical evidence that the SOCIO chatbot improves usability and (2) direct suggestions from users, as a starting point for understanding the impact of three human-computer interaction (HCI) usability characteristics (effectiveness, efficiency and satisfaction) that affect collaborative modelling tool and chatbot design.

Paper organization. In Sect. 2, we present the related work in usability experiments for chatbots. In Sect 3, we describe the design of the family of experiments, show the data analysis and results of this family of experiments. The paper finishes with the threats to validity section (Sect. 5) and discussion and conclusions (Sect. 6).

II. RELATED WORK

In [10], we reported a wider systematic mapping study (SMS) to identify the state of the art with respect to chatbot usability and applied HCI techniques in order to analyse how to evaluate chatbot usability. We concluded that chatbot usability is an incipient field of research, where the published studies are mainly surveys, usability tests, and rather informal experimental studies. Hence, it is necessary to perform more formal experiments to measure user experience and exploit these results to provide usability-aware design guidelines. We then updated the SMS, focusing on papers published from November 2018 to June 2020 applying the same procedure and search string as in [10]. In particular, we reviewed chatbot usability evaluation experiments to discover the recent trends and methodologies in the experimental software engineering field. Based on Ren et al.’s selection criteria [10], we also included papers describing controlled chatbot usability experiments and we excluded papers reporting only an evaluation or a quasi-experiment related to chatbot usability.

Finally, we retrieved ten primary studies ([11]-[20]) reporting experiments on the usability of chatbots which we used in this study. Only one study, designed as a within-subjects mixed-method experiment with different participant backgrounds, carried out replications of experiments [16]. Satisfaction continues to be the most popular usability characteristic, since it was evaluated more often. Task completion time and task completion are the efficiency and effectiveness characteristics attracting most interest, respectively. Within the primary studies, most chatbots are used as personal assistants [12][16][17][18][20].

* Corresponding Author.

DOI reference number: 10.18293/SEKE2021-043
Nevertheless, none of the chatbots were applied as modelling tools like SOCIO.

So far there have been three studies of the usability of SOCIO: the baseline experiment of this paper [11], and two separate evaluations [2][21]. All of these studies used questionnaires, and all participants had a SE background. Two small-scale evaluation experiments for SOCIO (with 19 and 8 participants) were reported in [2][21]. They measured SOCIO chatbot applicability for building an e-commerce class diagram in 15 minutes [2] and a consensus mechanism for choosing different modelling alternatives, where subjects had to choose the best of three options for two projects [21]. However, these studies focused on evaluating SOCIO separately using simple tasks. An evaluation experiment with a larger number of subjects (54 participants) comparing the SOCIO chatbot with the web-based application Creately was reported in [11]. Even though the subjects had identical backgrounds, session and task were confounded, highlighting the potentially detrimental effects of combining experimental results.

With the aim of moving beyond the limitations of the above studies, we build a family of experiments - which is defined as a group of at least three experiments with the same goal - by means of replication. Families of experiments allow surpassing the limitations in terms of sample size of individual experiments, and also, evaluating the effects of the treatments under different settings [22]. Families provide certain advantages for evaluating the effectiveness of SE treatments [4][5]: (i) because access to the raw data is granted in families, researchers can apply consistent pre-processing and analysis techniques to analyse the experiments, and, in turn, increase the reliability of joint conclusions; (ii) researchers conducting families may opt to reduce the amount of changes made across the experiments with the aim of increasing the internal validity of joint conclusions; and (iii) because families do not rely on already published results, joint conclusions are not affected by the detrimental effects of publication bias. Due to the advantages of families of experiments, we followed this approach to conduct our research.

III. FAMILY DESIGN

Since our family contains a SE baseline experiment and two replications, we designed the experiment according to the guidelines proposed by Santos et al. [22].

A. Objectives, Hypotheses and Variables

The objective of our family of experiments was to evaluate the usability, in terms of efficiency, effectiveness, and satisfaction, of the SOCIO chatbot through comparison in controlled experiments with the Creately web tool. The null hypotheses governing this research question is: H.x.0 There is no significant difference in EFFICIENCY | EFFECTIVENESS | SATISFACTION with respect to class diagram construction using SOCIO or Creately. This hypothesis is broken down into three specific null hypotheses, one for each usability characteristic (where x represents 1. Efficiency, 2. Effectiveness and 3. Satisfaction).

The main independent variable across all experiments is the modelling tool. The treatments are the SOCIO chatbot and the Creately web application. The response variable within the family is usability. Based on definitions of usability in ISO/IEC 25010:2011 [23], ISO 9241-11:2018 [24] and ISO/IEC/IEEE 29148:2018 [25], and Hornbæk’s guide [26], efficiency, effectiveness and satisfaction are commonly measured attributes for evaluating product usability. In view of this, we measure usability as efficiency, effectiveness and satisfaction.

We measured efficiency in terms of speed and fluency. Speed corresponds to the time taken to complete the tasks. Fluency corresponds to the number of discussion messages exchanged between the teammates during task development via the Telegram group. We measured effectiveness as completeness, based on the perceived success of each class diagram compared with the ideal class diagram that we built to measure the solutions produced by all participants [11][26]. In particular, the speed, fluency, and completeness metrics refer to social complexity and sociability and are typically evaluated when measuring macro-level usability (tasks requiring hours of collaboration) [23][24][26]. To assess and quantify satisfaction, we modified the System Usability Scale (SUS) questionnaire [21][27] to suit our experiments. Ease-of-use and learnability are two measured sub characteristics included in SUS questions [26]. There are ten SUS questions –each question is scored on a five-point Likert scale– and four open-ended questions. Finally, we adopted Brooke’s equation [27] to derive the numerical value of each participant’s satisfaction score. The median of the scores given by all three members of each team –to each question– is selected as the team score.

B. Design of the Experiments

All three experiments in our family have an identical experimental design. The study employed a two-sequence and two-period within-subject crossover design (see Table I). We chose a crossover design to avoid the influence of the period on the treatment and assure that there was no learning effect between the two periods [28].

<table>
<thead>
<tr>
<th>Group</th>
<th>Period 1 (Task 1)</th>
<th>Period 2 (Task 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1 (SC-CR)</td>
<td>SOCIO</td>
<td>Creately</td>
</tr>
<tr>
<td>Group 2 (CR-SC)</td>
<td>Creately</td>
<td>SOCIO</td>
</tr>
</tbody>
</table>

The participants were grouped into three-member teams, where each team was considered as a subject. We put the full participant name list in a random team generator (www.randomlists.com/team-generator) to generate teams. All teams were assigned to either of two groups (Group 1 or Group 2), where each group applied the treatments in a different order. Participants did not receive any training and signed an informed consent before the experiment. After a 10-minute tutorial on the tool that they were to use in each period, they were required to perform the task in 30 minutes. Group 1 implemented Task 1 using SOCIO in the first period followed by Task 2 with Creately in the second period (i.e., SC-CR sequence).

On the other hand, Group 2 implemented Task 1 with Creately first, followed by Task 2 with SOCIO (i.e., CR-SC sequence). Task 1 was to develop a class diagram representing a store, including the management of products and customers. Task 2 consisted of designing the class diagram of a school to support courses and students. At the end of each period, all participants filled in a modified and validated SUS questionnaire. We did not ask participants which tool they preferred until the end of the second period.
C. Subjects

Participants were recruited from two universities in two countries: (1) the Universidad de las Fuerzas Armadas ESPE Extensión Latacunga (ESPE-Latacunga) in Ecuador (UNIV-1), and (2) the Escuela Politécnica Superior of the Universidad Autónoma de Madrid (EPS-UAM) in Spain (UNIV-2), all participants are undergraduate students who were completing a degree in Computer Engineering. Each participant only participates once. A total of three experiments were run. 18 subjects (54 participants) of the baseline experiment (EXP1) are from UNIV-1. 10 subjects (30 participants) of the second experiment (EXP2) are from UNIV-2. The third experiment (EXP3) contains 11 subjects from UNIV-2 and 5 subjects from UNIV-1 (48 participants in total). The subjects were selected using convenience sampling: participants were students of academic staff teaching SE-related courses, and all participants volunteered to participate. All participants were required to complete a pre-test questionnaire that assessed demographic data and related experience and knowledge.

As Fig. 1 shows, average subject experience appears to be slightly heterogeneous, but the gaps between each experiment appear to be small (i.e., never greater than 1). Although 37% of participants have no experience in using Telegram, they are regular social media users. This ensures that they can complete the task since no complicated operations are required. However, the inclusion of subjects with no previous experience with chatbots does pose a threat to the validity of the results. Despite the fact that none of the participants were native English speakers, they all claimed to have at least an intermediate level of English. As there are no significant differences between the three experiments in terms of age, gender, knowledge background, social media usage habits, smartphone or tablet ownership, we consider that the participants across the countries are comparable.

IV. RESULTS AND DATA AGGREGATION

A. Analysis Approach

In response to the research question, we follow Santos et al.’s guidelines [22] to analyse the family of experiments. For each metric, we provide: (i) a profile plot showing the mean effect of the treatments across the experiments (ii) a violin-plot and the descriptive statistics divided by treatment and by experiment; and (iii) the joint results of all the experiments together applying a one-stage individual participant data (IPD) meta-analysis, reporting the contrast between treatments as an extra parameter in the linear mixed model (LMM) model to account for the difference between results across experiments [22][29]. The profile-plots give a bird’s eye view of the data at family level and check for the existence of patterns across the results [22]. The descriptive statistics and violin-plots ease the understanding of the data in each experiment. We followed an IPD meta-analysis approach rather than a meta-analysis of effect sizes, because we had access to the raw data of the experiments [29].

As all the experiments have an identical (i.e., a cross-over) design, we analyse them following Vegas et al.’s advice [28]. In particular, we analyse the experiments using linear mixed models (LMMs) [28]. We used LMMs rather than their non-parametric counterparts because: (i) commonly used non-parametric models are not useful for studying the effect of multiple factors at the same time (e.g., period, treatment, and sequence on the outcomes); (ii) the overall sample size (i.e., 44 teams, each with two data-points —one per session, a total of 88 data points) may suffice to make the central limit theorem hold [30], and thus, interpret the results despite data non-normality.

In particular, we fit a three-factor LMM [31] for each metric: period (i.e., 1 or 2), treatment (i.e., SOCIO, or Creately), and sequence (i.e., SOCIO-Creately or Creately-SOCIO). We add an extra parameter to the LMM to account for the difference between results across the experiments (i.e., Experiment), which is a common feature of stratified individual participant data (IPD) models [22]. We interpret the statistical significance of the results with the corresponding ANOVA table of LMMs.

B. Response Variables

1) Efficiency

As Fig. 2 and 3 and the descriptive statistics (Table II) show, the aggregate time appears to be less for SOCIO than for Creately in two out of three of the experiments. The difference in performance between the treatments is statistically significant in the ANOVA table (Table IV). According to the pairwise contrast between the treatments in Table V, the participants took an average of 1.14 minutes longer with Creately than with SOCIO.
and EXP2. The opposite applies to EXP3, albeit to a lesser extent. As Table IV and Table V show, the difference in satisfaction scores appears to be significant at the 0.1 level. In other words, participants appear to have higher satisfaction scores with SOCIO.

As we can see in the plots and the descriptive statistics (Figs. 4 and 5 and Table II), the participants tend to send more messages with Creately than with SOCIO. Besides, as Table V shows, the difference in the number of messages is statistically significant. In particular, the participants send up to 7.23 more messages with Creately than with SOCIO, as shown in Table V.

Considering different textual communication styles, we treat a complete single sentence or an emoji as a message. Although message exchange was encouraged, we considered a low number of messages is an indicator that fewer communication efforts were required, since users were immediately able to observe the changes in the class diagram.

2) Effectiveness

As we can see in Figs. 6 and 7 and Table III, completeness appears to be similar for both tools. Besides, as shown in Table IV and Table V, the observed difference in completeness (-0.0033) was negligible and not statistically significant. In sum, Creately and SOCIO appear to perform similarly in terms of completeness.

3) Satisfaction

As Figs. 8 and 9 and Table III show, the participants appear to be more satisfied with SOCIO than with Creately in EXP1 and EXP2. The opposite applies to EXP3, albeit to a lesser extent. As Table IV and Table V show, the difference in satisfaction scores appears to be significant at the 0.1 level. In other words, participants appear to have higher satisfaction scores with SOCIO.

TABLE II. DESCRIPTIVE STATISTICS FOR EFFICIENCY. LEGEND: TR=TREATMENT; CR=CREATELY; SC=SOCIO; FLUEN=FLUENCY

<table>
<thead>
<tr>
<th>Metric</th>
<th>Exp</th>
<th>TR</th>
<th>Team</th>
<th>Mean</th>
<th>Std. Dev.</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPEED</td>
<td>EXP1</td>
<td>CR</td>
<td>18.00</td>
<td>28.83</td>
<td>1.76</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>EXP1</td>
<td>SC</td>
<td>18.00</td>
<td>27.06</td>
<td>2.62</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>EXP2</td>
<td>CR</td>
<td>10.00</td>
<td>27.18</td>
<td>2.69</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td>EXP2</td>
<td>SC</td>
<td>10.00</td>
<td>25.30</td>
<td>2.63</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>EXP3</td>
<td>CR</td>
<td>16.00</td>
<td>29.19</td>
<td>1.42</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>EXP3</td>
<td>SC</td>
<td>16.00</td>
<td>29.19</td>
<td>1.56</td>
<td>30</td>
</tr>
<tr>
<td>FLUEN</td>
<td>EXP1</td>
<td>CR</td>
<td>18.00</td>
<td>19.56</td>
<td>16.30</td>
<td>13.50</td>
</tr>
<tr>
<td></td>
<td>EXP1</td>
<td>SC</td>
<td>18.00</td>
<td>9.61</td>
<td>11.51</td>
<td>5.00</td>
</tr>
<tr>
<td></td>
<td>EXP2</td>
<td>CR</td>
<td>10.00</td>
<td>75.40</td>
<td>40.84</td>
<td>68.00</td>
</tr>
<tr>
<td></td>
<td>EXP2</td>
<td>SC</td>
<td>10.00</td>
<td>57.00</td>
<td>19.98</td>
<td>51.00</td>
</tr>
<tr>
<td></td>
<td>EXP3</td>
<td>CR</td>
<td>16.00</td>
<td>63.00</td>
<td>45.85</td>
<td>70.00</td>
</tr>
<tr>
<td></td>
<td>EXP3</td>
<td>SC</td>
<td>16.00</td>
<td>65.81</td>
<td>46.00</td>
<td>66.00</td>
</tr>
</tbody>
</table>

As we can see in Figs. 4 and 5 and Table II, the participants tend to send more messages with Creately than with SOCIO. Besides, as Table V shows, the difference in the number of messages is statistically significant. In particular, the participants send up to 7.23 more messages with Creately than with SOCIO, as shown in Table V.

Considering different textual communication styles, we treat a complete single sentence or an emoji as a message. Although message exchange was encouraged, we considered a low number of messages is an indicator that fewer communication efforts were required, since users were immediately able to observe the changes in the class diagram.

2) Effectiveness

As we can see in Figs. 6 and 7 and Table III, completeness appears to be similar for both tools. Besides, as shown in Table IV and Table V, the observed difference in completeness (-0.003) was negligible and not statistically significant. In sum, Creately and SOCIO appear to perform similarly in terms of completeness.

3) Satisfaction

As Figs. 8 and 9 and Table III show, the participants appear to be more satisfied with SOCIO than with Creately in EXP1 and EXP2. The opposite applies to EXP3, albeit to a lesser extent. As Table IV and Table V show, the difference in satisfaction scores appears to be significant at the 0.1 level. In other words, participants appear to have higher satisfaction scores with SOCIO.
results [22][32]. We also evaluated the quality of the constructed class diagrams with respect to different aspects so as to give a better understanding for the time metric. In order to ensure the transparency of the results, we provide the original data, statistical analysis carried out and collaboration examples with chatbot SOCIO and Creately in the supplementary materials at https://bit.ly/34v7OTs.

Fig. 8. Profile-plot for satisfaction

Fig. 9. Violin-plot for satisfaction

Unacknowledged variables confounded with the investigated variable may pose threats to internal validity. Since subject background (e.g., different universities) is another potential independent variable, this threat may compromise the validity of the results. In EXP3, we tried to mitigate this threat by conducting the experiment in the same universities as EXP1 and EXP2.

In terms of construct validity, we acknowledge that self-assessment questions may not properly reflect the knowledge background of the participants because they may not be able to honestly assess either their knowledge level or their characteristics—even if they used chatbot/social network as frequently as each other. This fact add bias on the response variable of satisfaction.

A probable external threat is the generalization of results. As usual in SE experiments [33], we had to rely on toy tasks to evaluate and compare the performance of two tools. Having said this, the subjects that make up this family of experiments are computer science students with sufficient knowledge of the field. In view of this, our findings are limited to academia and are not generalizable to industry.

VI. DISCUSSION AND CONCLUSIONS

To the best of our knowledge, there is no other chatbot offering a similar service to SOCIO. Although SOCIO chatbot usability was evaluated in two other small-scale evaluation results [2][21] previously, the number of subjects was smaller than provided by this family of experiments and it was not compared with other tools. Our family of experiments is the first and only research to evaluate the usability of the SOCIO chatbot comprehensively with regard to effectiveness, efficiency and satisfaction. Particularly, this family consolidates the previous result of the baseline experiment [11] thanks to a bigger sample size and more powerful statistical results. The information aggregated at family level is much more accurate than for individual experiments, which, in many cases, are unable to observe the existing differences. For instance, the treatment is not statistically significant for all variables in EXP3, which is not the case in the family of experiments (see supplementary material).

We followed a mixed method to provide joint results and identify variables impacting results. With our family of experiments, we observed that subjects take longer and send a larger number of messages to build class diagrams with Creately than with SOCIO. In other words, SOCIO outperforms Creately in terms of efficiency. Regarding effectiveness, results are similar for both tools. For satisfaction, we can conclude that participants were more satisfied with SOCIO than with Creately.

In addition, with the aim of identifying concrete opinions related to the satisfaction from subjects, we extended SUS questionnaire (see supplementary material) with four open-ended questions (concerning positive and negative aspects of the two tools, suggestions and user preferences) in order to gather definite satisfaction-related opinions from subjects. By analysing responses to open-ended questions, we find some insight as follows. Many participants remarked that they found both tools to be satisfactory in terms of responsiveness, ease of use, and collaboration capabilities. Creately was praised for its friendly interface. SOCIO was more fun to use. Quite a few participants complained about the SOCIO chatbot help web page, whereas the biggest problems with Creately were related to real-time collaboration, which produced some errors when loading on some of the user’s computers.

This research contributes to the empirical analyses of the evaluation of chatbot usability, in particular, the chatbot SOCIO. There is the existence of statistically significant differences with medium effect size. Additionally, our experiments provide further information for developers regarding the usability evaluation of SOCIO chatbot and Creately. We conclude that chatbots may aid in the creation of class diagrams. In particular, their speed may be valuable, especially in view of the satisfaction shown by the participants with their use.

Future studies will focus on investigating this updated versions of the SOCIO chatbot. Accordingly, it is possible to clarify the required evidence-based SOCIO chatbot improvements. Currently, work is underway to develop four different updated versions of the SOCIO chatbot: (1) Provide different help when the SOCIO chatbot does not understand the user well according to a different situation. (2) Add functionalities requested by users: users will be able to delete any elements that they like by clicking the buttons underneath, and users will be able to choose how many steps to cancel or redo at a time instead of deleting or redoing one by one. (3) Provide option to select the appearance of the class diagrams. (4) Update and supplement the help page for all three versions.

ACKNOWLEDGEMENTS

This research was funded by the Spanish Ministry of Science, Innovation and Universities research grant PGC2018-097265-B-I00 and MASSIVE project (RTI2018-
REFERENCES


Evaluation of Chatbots Usability Experimentation

Ranci Ren
Dep. Ing. Informática
Univ. Autónoma de Madrid
Madrid, Spain
ranci.ren@estudiante.uam.es

John W. Castro*
Dep. Ing. Informática y Ciencias de la Computación
Universidad de Atacama
Copiapó, Chile
john.castro@uda.cl

Silvia T. Acuña
Dep. Ing. Informática
Univ. Autónoma de Madrid
Madrid, Spain
silvia.acunna@uam.es

Abstract—Context: The interest in developing chatbots is on the rise as the usability evaluation is an essential step in the chatbot development process; the number of experimental studies of chatbot usability has grown as well. Objective: Aggregating and concluding the features and metrics used to evaluate the usability of chatbots in experiments, to identify the state of the art of chatbots usability experimentation. Method: A systematic mapping study has been conducted, searching in five scientific databases. Results: Of 363 papers, 14 papers with experiments were selected as the primary studies. The published works in this area were initiated in 2018. Control tools are applied commonly in experiments. Various advantages and shortages of chatbot usability experiments were revealed, for example, most of the experiments do not provide raw data, and only one of the identified works replicated the experiment. Conclusions: An increased interest in usability experimentation of chatbots is observed in recent years. The chatbot usability experiment should be more replicable to improve the reliability of experimental results.

Keywords—Usability, Chatbots, Family of Experiments

I. INTRODUCTION

A chatbot, also known as a chatterbot, is a domain-specific text-based software that supports human users with specific services [1]. The remarkable advancement of natural language processing and machine learning is causing a seismic shift, in that sense, this created unlimited possibilities, productive and useful experiences through chatbots who can access and interact with digital services in many different applications [2][3]. Compared with other communication channels (e.g., e-mail), not all users are willing to fully trust the chatbot due to understanding ability and response quality, chatbot design is still far from reading users’ minds, in this context, it is necessary for better integration between usability evaluation and the chatbot [4]. Usability evaluation refers to how well users can learn and use software to meet their requirements and refers to how satisfied users are during the process [5]. In software engineering (SE), usability has been recognized as a crucial quality characteristic for success in the competitive commercial world [7]. The choice of evaluation methodology must be applied appropriately for the current research question or issue [5]. Apparently, usability evaluation of chatbots is not a mature field so far [4]. In general, usability evaluation of chatbots learns and borrows experience from experimentation in software engineering (ESE). We noticed that the families of experiments are being run in increasing numbers in ESE [8]. It is the unanimous opinion of the scientific community that the veracity of the base experiment results can only be established by replication and contrast of results [9]. A family of experiments is a set of experimental replications with the same goal [8]. The families of experiments allow to obtain a greater statistical power due to the greater number of involved subjects [10], increase the internal validity of joint conclusions and the reliability of the findings. Due to the strengths of families of experiments, we pay special attention to the adoption of families of experiments in chatbot usability evaluation. To explore the chatbot usability experimentation, we did a preliminary investigation, and we did not find any previous study or literature review that could bring us a consolidated view. As described by Ren et al. [4], we noticed that chatbots and their relevant usability evaluation had become prevalent themes and the number of publications started to grow from the year of 2015; however, they did not pay attention to the usability experiment of chatbots. For this purpose, we conducted a systematic mapping study (SMS) on top of a baseline study [4] with the aim of (i) explore the state-of-the-art on chatbots usability experimentation, (ii) identify the metrics used in experiments to measure chatbot usability in SE. The nature of our contribution is providing a map of what has been published since we have included all reported references in the literature of our SMS on chatbot usability experimentation. With this information, researchers interested in conducting experiments and/or replications related to the usability of chatbots will obtain a baseline of aspects that they should consider.

Paper organization. In Sec. 2, we outline the research method of the SMS. In Sec. 3, we provide the answer to each of the research questions. In Sec. 4, we discuss the results and threats to validity. Finally, we outline the conclusions of our study in Sec. 5.

II. RESEARCH METHOD

The secondary study reported in this paper has been developed following the guidelines established by Kitchenham and Charters [11].

Objectives and Research Questions. The main objective of this study was to map the usability experiments of chatbot in aspects of publication status, and measured metrics in experiments. This gave rise to our research questions: (RQ1) What is the state of the art of chatbots usability experimentation? (RQ2) How to evaluate the usability of chatbots in experiments?

Search String Selection. We first piloted various synonymic search strings. The rationale behind the selection of our final search string is that it returns the most records, and the results are more balanced between the different databases. Our final search string was: "usability OR "usability techniques" OR "usability practice" OR "user interaction" OR "user experience") AND (chatbots OR "chatbots development" OR "conversational agents" OR "chatterbot OR "artificial conversational entity" OR "mobile chatbots").

Databases and Search Protocol. The IEEE Xplore, ACM Digital Library, SpringerLink, Scopus and ScienceDirect academic databases (DBs) were used in the SMS process. The selection criteria used to retrieve the fundamental studies are

* Corresponding Author.
DOI reference number: 10.18293/SEKE2021-053
summarized below. We dismissed an article whenever at least one of the exclusion criteria was met. Inclusion criteria: (the abstract or title mentions an issue regarding the chatbots and usability) OR (the abstract mentions an issue related to usability engineering or HCI techniques) OR (the abstract mentions an issue related to the user experience) AND (the paper describes the experiment of chatbot usability). Exclusion criteria: (the paper does not present any evaluation or experiment related to chatbot usability) OR (the paper does not present any issue related to the chatbots and usability) OR (the paper does not present any issue related to the chatbots and user interaction) OR (the paper does not present any issue related to the chatbots and user experience) OR (the paper is written in a language other than English).

Search Process. We reviewed works about the experiments of chatbot usability, which were published from 2014 to June 2020. Once we identified the search strings and defined search fields, we started our search process. A total of 363 Retrieved Papers were found in the different DBs. Then the duplicate papers were removed from the retrieved papers, 323 papers were filtered to the group of Non-Duplicate Retrieved Papers. A peer review was carried out on these 323 papers applying the inclusion and exclusion criteria to the title and abstract. Discrepancies were resolved through a discussion. As a result, we obtained 86 Candidate Papers. To determine if candidate papers have relevance regarding the usability of chatbots and the execution of the chatbot usability experiment, we reviewed each candidate paper again, applying the inclusion and exclusion criteria. However, this time we especially reviewed the full text. The results were cross-checked by two HCI experts. Finally, 14 papers formed the Experiment Papers used in this study. The results of selection were assessed by two HCI experts, each disagreement has been discussed and resolved during the meeting. The remained 14 experiment papers for the analysis and extraction of the results are shown in Appendix A.

III. RESULTS

RQ1: What is the state of the art of chatbots usability experimentation? The raw data were poorly reported among 14 experiments as only one experiment provided access to their raw data in the paper. As shown in Fig. 1, a synthetic view of the identified primary studies, the results have been segmented into two areas. The left-side consists of two scatter (XY) plots (top and bottom) with bubbles at the junctions of the year-type of publication categories (left side - top) and usability feature-type of publication categories (left side - bottom). With regard to the types of publications, 50% of publications are conference papers, 28.6% are journal article, 21.4% are chapters book. The size of each bubble was determined by the number of experiment papers that had been classified into each category. The right-side of Fig. 1 presents the number of primary studies published per year. As can be seen from the upper right part of Fig. 1, the interest in chatbots usability experimentation is increasing and is very recent; initial works are from 2018. Considering that the search was carried out until June 2020, the number of identified works in our SMS for 2020 is high. Satisfaction is the most widely measured usability feature. Note that the number of papers in the lower part of Fig. 1 does not match the number of papers in the upper part. The reason is that the same paper can discuss several usability features. In aspect of the types of chatbots, most chatbots are deployed as the personal assistant [PS2][PS4][PS6][PS10][PS11][PS13], especially in the health care domain [PS5][PS7][PS14], some act as e-commerce tools [PS9][PS12], collaborative tool [PS8] and recommender [PS3].

The Replication of Experiments. Upon the usability experiments we reviewed, there is only one study presented by Huff-Jr et al. [PS6] conducted a replication of an experiment with consistent experimental design but different participant background. They used a within-subjects mixed-method design, and they analyzed data by analyzing qualitative contents and a multilevel linear model. The total sample size of replications is 35, although the authors do not report the corresponding sample size of each replication. To the best of our knowledge, a family of experiments should include at least three experiments [8], while a single experiment had been replicated in their work—that is, it forms a set of two experiments since two experiments are able to aggregate the data to evaluate the effect of chatbots—we classified them as a family of experiments.

Sample Sizes. Although the sample size varies in different usage and development phases, as the recently published experiments have, the sample sizes of usability experiments for chatbots are relatively small. Of the 14 experiments, 50% of experiments contained less than 30 subjects, 28.6% contained between 30 and 50 subjects, and 14.3% contained between 100 and 500 subjects. One experiment did not detail the sample size [PS11].
Types of Subjects. 35.7% of the experiments include students, while most of the researchers did not limit academic background and grade. There were 29% that included experienced users or experts and company employees. Three experiments included farmers, children, and residents, respectively. Two experiments did not detail the types of subjects. Only one experiment used compared group, graduates and undergraduates [PS8].

Experimental Design and Procedure. 71.4% of experiments were defined as a within-subject design. Since the sample sizes of identified experiments are relatively small, the within-subject design has better statistical power by doubling data points. In SE, experimental design plays a role in controlling for extraneous variables: mature experiments are run with pre-established protocols defining the experimental settings and the set of procedures that must be strictly adhered to during the execution and analysis of the experiments. By contrast, many usability experiments of chatbots are formed without any a priori plan or experimental design definition. Furthermore, the prior experience and technical knowledge have an impact on the global usability of Conversational Agents [PS13], while the pre-user experience or knowledge related to chatbot seems didn’t be measured during some experiments [PS1][PS11].

Statistical Techniques. Statistical techniques are categorized from two perspectives: statistical descriptions and statistical inference. The statistical descriptions (Table II) are representation methods that integrate multiple datasets in a visual way to give context to the data and to improve reader understanding. There is an experiment that has not yet been executed [PS11]. Among 13 experimental results of chatbots usability, box plot and descriptive statistics were the most used presentation formats. Statistical inference was used to analyze 11 experiment results. 7 experiments used parametric tests [PS2][PS5][PS7][PS9][PS12][PS13][PS14], and 4 experiments used non-parametric tests [PS1][PS3][PS5][PS9]. The majority of the authors did not explain the motivation behind adopting the technique or indicate the challenges or advantages of adopting the technique.

<table>
<thead>
<tr>
<th>TABLE II. STATISTICAL DESCRIPTIVE REPRESENTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistical Descriptive Representation</td>
</tr>
<tr>
<td>Box plot</td>
</tr>
<tr>
<td>Descript. statistics table</td>
</tr>
<tr>
<td>Histogram</td>
</tr>
<tr>
<td>Line chart</td>
</tr>
<tr>
<td>Scatter plot</td>
</tr>
<tr>
<td>Textual description</td>
</tr>
</tbody>
</table>

IV. DISCUSSION AND VALIDITY THREATS

Although our goal is to present an analysis for chatbot usability experimentation, we noticed that the interfaces of most current chatbots take a form of an NL dialog: the development of chatbots has become standardized because there are many build platforms for different goals and usages that have been widely used [PS1][PS6][PS10]. Of the initial 363 papers selected in well-known electronic research databases [14], studies were selected following a rigorous process, from selecting studies to solve disagreements found during the selection process. The comparison of two or more treatments and randomization of subjects are our key points to identify if the study described an experiment [12] when we reviewed each paper. The usability experiment of chatbot correlates to chatbot development; however, there is only one experiment related to a usability experiment of chatbot in an advanced or modified version [PS12]. To obtain reliable experimental results, all aspects of treatment (except for the manipulation of factors) should remain similar across all groups, as irrelevant poses a threat to validity. We noticed that many studies did not clearly state extraneous variables control in their experiment designs. For example, they did not discuss the possible learning effects between different sessions [PS6][PS10]. We observed that most chatbots were experimented based on some specificities—including the relatively small sample size, the subjects with a specific background, the tasks being preset, and whether it was the users’ first encounter with a chatbot—as the expansion of experimental results to the industrial field is fairly limited. Besides, there is a work that did not published the experimental results as of our search date [PS11].

The first threat to the validity of this work is the bias in the paper selection process. Although the selection criteria and results have been double-checked and accepted by other authors, the publications were evaluated and classified based on the judgment and experience of the authors, and other researchers may have evaluated the publications differently. The second point is related to the type of studies included in this work. We expanded the search scope by using search strings that identify a wider range of publications. On the one hand, this SMS was developed using five databases as they were considered the most complete and most used database in SE. On the other hand, this search includes only papers written in English. Nonetheless, relevant papers produced by additional databases or resources or written in other languages could have overlooked.

V. CONCLUSION AND FUTURE WORK

RQ1: What is the state of the art of chatbots usability experimentation? From our SMS perspective, chatbots usability experiments are being run in increasing numbers (see Fig. 1). With regard to publication venue, half of the reviewed papers in our SMS are published through conferences. We notice that control tools are applied commonly in experiments, most studies used real-life products as control tools [PS1][PS2][PS5][PS8]. To determine whether the chatbot was able to provide a similar experience to the user, some developed diverse version of chatbots with different functions or expression [PS3][PS9][PS10].

We also observed that many experiments did not define the research question or hypothesis follow ESE methods [12], or the proposed research questions are related to usability but are not limited to usability. In general, most studies investigate not only usability factors but also the quality of the interaction or chatbot performance [PS3][PS7][PS8][PS10], in order to understand the chatbot usability comprehensively and also some studies investigate the relationships between the usability and other factors (e.g., acceptability) [PS5][PS10][PS14]. The majority of the experiments did not provide access to raw data. This situation prevents rigorous peer-review and does not allow third-party researchers to reanalysis using aggregation methods that may be more appropriate than the original one [8].

RQ2: How to evaluate the usability of chatbots in experiments? We notice: (i) the questionnaire is the most used usability technique; (ii) the family of experiments was barely
used in this field so far since only one experiment contained replications was found; (iii) the within-subject design is the most popular design on chatbots usability experimentation; (iv) 50% of the experiments included a small sample size (less than 30 subjects) and the most subjects are students; (v) the number of tasks is relatively small, as most of the experiments applied less than six tasks; and (vi) parametric tests were the most used inference to analyze the experimental result in experiments.

We suggest that the researchers: (i) provide access to full raw data to guarantee the replicability of the experiment and transparency of results; (ii) consider the family of experiments or conduct replications of the baseline experiment to consolidate the experimental result and to increase the statistical power; (iii) more third-party evaluations should be considered in chatbot usability evaluation, as they do not suffer from the bias introduced in the previous development process. Considering that the work is limited by search date, databases and search strings, this study could be replicated in a future study. Based on this research results, we plan to conduct a family of experiments to evaluate a chatbot's usability with an advanced version to fill the gap and explore the topic further.

ACKNOWLEDGMENT

This research was funded by the Spanish Ministry of Science, Innovation and Universities research grant PGC2018-097265-B-I00 and MASSIVE project (RTI2018-095255-B-I00). Also, it received support from the Madrid Region R&D programme (FORTE project-P2018/TCS-4314).

APPENDIX A: PRIMARY STUDIES


REFERENCES


From word embeddings to text similarities for improved semantic clustering of functional requirements

Takwa Kochbati, Sébastien Gérard, Shuai Li, Chokri Mraidha

Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
{takwa.kochbati, sebastien.gerard, shuai.li, chokri.mraidha}@cea.fr

Abstract— Requirements engineering starts by requirements elicitation which consists in gathering software requirements from stakeholders. Then, the elicited requirements are usually manually recorded in a requirements specification document. In recent years, modern software projects are becoming more complex than projects of the past due to the increase in the number of requirements and stakeholders involved in a project. Thus, manually managing requirements becomes a tedious, time consuming and error-prone task. One historical strategy to manage this kind of complexity is “divide to conquer”, meaning to categorize them into groups in order to breakdown the system into a set of smallest sub-systems at early stages. In this paper, we propose an approach to automatically cluster functional requirements based on their semantic similarity which is the usual strategy used by system architects to define sub-systems candidate to simplification of the original problem. First, we use word2vec, as a predictive word embedding model to compute the word-level similarity. Second, we derive the requirement-level similarity using a scoring function for text similarity. Third, we adopt hierarchical clustering to group the requirements. Experimental results performed on four open-access software projects show that our approach succeeded to improve the results of clusters identification compared with existing studies.

Keywords-component; software requirement; clustering; word embedding; natural language processing.

I. INTRODUCTION

Requirements elicitation is the first step in developing a software product. In this step, engineers discover and collect requirements from customers and then, they manually record them in a requirements specification document. The gathered requirements describe different aspects of the target software in natural language and they are mainly classified into functional and non-functional requirements [1]. Functional requirements describe the functional behavior and the features of the software system while non-functional requirements define the system attributes such as performance, security, reliability as well as the system operational conditions such as power consumption and environmental conditions.

Requirements elicitation has a significant impact on information systems quality and success, as the errors introduced at the beginning stages of development are the hardest and most expensive to correct [2].

Hence, it is crucial that the requirements set has to be well understood and well managed by engineers [3].

System design constraints evolves more and more requiring to embed more stakeholders in the projects to handle various new concerns - such as security, safety, cost, and sustainability – earlier in the process, at specification time. Consequently, modern software projects are becoming many times larger and hence more complex than in the past. Especially, the exponential growth of the number of requirements raises difficulties in managing manually the requirements and having a clear crystal view of the expectation and scope of the system to be designed [4]. One of the most used and efficient design paradigms to deal with complexity is the well-known “divide-to-conquer” strategy i.e., building smallest pieces to reduce the complexity. Herein lies the importance of an automatic solution to categorize software requirements into a set of groups in order to breakdown the target software system into a set of smallest sub-systems at early stages of the development process.

In this paper, we propose a clustering solution to automatically group functional requirements based on their semantic similarity. We use and analyze the semantic information of the requirements to compute the requirements similarity at two levels: at the word level, but also at the statement level of the requirements (i.e., local versus global semantics of the requirements). In that context, we make the following contributions: 1) we use a neural word embedding model, word2vec, as a predictive model to compute the word-level similarity; 2) then, we derive the requirement-level similarity using a scoring function for text similarity computation; 3) finally, we adopt hierarchical clustering combined with a pre-defined criteria to group the requirements in specific clusters. To evaluate our proposal, we have successfully applied it to four open-access software projects.

The remainder of the paper is structured as follow: section 2 discusses the related works; section 3 describes the proposed approach; section 4 provides the experimental evaluation settings; section 5 provides the results analysis; section 6 raises the limitations and the threats to validity and finally, section 7 concludes the paper.

DOI reference number: 10.18293/SEKE2021-056
II. RELATED WORKS

In recent years, the usage of clustering techniques in the early phases of software engineering has gained a lot of attention.

In [5], the authors developed a tool based on hierarchical clustering of requirements in order to propose a packaging solution for software engineers. They defined a similarity measure that aims to cluster classes with high number communication in the same package. The optimal number of clusters is manually selected by software engineers based on the hierarchical tree generated by the clustering algorithm.

In [6], the authors present an approach based on concepts clustering to visualize requirements at different levels of granularity. They employed word2vec as a predictive model to compute similarity between concepts.

The authors in [7] propose an initial clustering of responsibilities from requirements, in order to detect architecture components. The approach is validated using four different clustering algorithms and several validity metrics. The similarity function is computed according to the verb phrase each responsibility contains, and the direct object it is related to.

In [8], the authors present an approach to cluster and sequence user stories in order to assist software engineers in the implementation phase. They employed clustering algorithm and the silhouette score to identify the best clustering solution.

In [9], the authors propose an approach that clusters similar requirements in order to reuse them in software product lines (SPLs). They compared the performance of two clustering algorithms based on a distance measure in order to identify similar requirements.

In [10], the authors demonstrate the use of the HAC algorithm to group functional requirements based on their similarity. Their work aims to breakdown the project into a set of sub-projects at early stages. They use traditional vector space models (VSMs) to vectorize text requirements and use the cosine similarity to measure the semantic similarity between requirements.

All these techniques inspire our work. However, some of these approaches suffer from a lack of automation as for example when defining the optimal number of clusters [5], others rely on the similarity between words or concepts in each requirement [6], [7]. Moreover, many works rely on traditional distributional semantic models (DSMs), for instance Vector Space Model (VSM) [8], [10] and Latent Semantic Analysis (LSA) [9] to calculate the similarity. The main limitation of these techniques is that they are considered as “count” models as they rely on counting the co-occurrences among words by operating on co-occurrence matrices. Thus, sentences with similar context but different term vocabulary will not be considered as similar. Consequently, traditional DSMs usually achieve worse results than neural word embedding models, which can be seen as predictive models [11].

The main novelty of our proposal is that we benefit from using the neural word embedding model word2vec as predictive model, to compute word level similarity and then, derive the requirement level similarity using a scoring formula for text similarity.

III. THE PROPOSED APPROACH

In this section, we explain how our approach processes in order to generate automatically the clusters from natural language requirements as illustrated in the process shown in Figure 1.

In what follows, we detail the particular techniques used in each step.

A. Preprocessing

Preprocessing is the first step of the approach in which, the input functional requirements expressed in natural language are normalized through four steps: (i) tokenization, i.e., the decomposition of a sentence into a set of individual words; (ii) stop-words removal, i.e., the elimination of common English words; (iii) punctuation removal; (iv) stemming, i.e., the transformation of each word to its root (e.g: “adding” becomes “add”).

B. Semantic similarity computation module

The preprocessed requirements are then introduced into the semantic similarity computation module. Traditional approaches to compute the similarity between two text segments consist in using lexical matching method, and producing a similarity score based on the number of lexical units that occur in both input segments. However, these lexical similarity methods cannot always identify the semantic similarity of texts as they aim to determine whether the words
in two texts have similar spellings [12]. For example, the “US” would be closer to the “UK” this way, than it would be to the “States”.

Going beyond these traditional methods, we compute and analyze the semantic information at two levels: locally, for each word contained in a requirement description, but also globally at the statement level.

1) Word-level similarity computation

In order to compute the word-level similarity, one must rewrite the preprocessed requirements from natural language to a machine-readable and analyzable format. Thus, words should be transformed into numerical vectors that work with machine learning algorithms. To this end, we use the word2vec model, a two-layer neural network that is used to produce word embeddings (i.e., vectors).

The input of word2vec is a text corpus. Given enough text data and contexts, word2vec can achieve highly accurate semantics of the words appearing in the corpus and establish a word’s association with other words in the semantic space. Moreover, word embedding models have shown to outperform traditional DSMs which are considered as “count” models as they count co-occurrences among words by operating on co-occurrence matrices [11].

Since a word embedding model is supposed to be of high quality when trained with large corpus, we use the pretrained word2vec model on 100 million words of Google News dataset (https://code.google.com/archive/p/word2vec/). However, even if the used corpus is large (e.g., Google News), some domain-specific words found in the requirement statement may be unknown in the corpus. In this case, as suggested in [13], we assign a random vector to the missing word. Then, we compute the semantic similarity between each pair of the obtained word vectors belonging to two different requirement statements using the cosine similarity measure. The cosine similarity principle consists in computing the cosine of the angle between two word vectors. Thus, the cosine similarity of two similar words vectors is close to 1, and close to 0 otherwise.

2) Requirement-level similarity computation

After obtaining the word-level similarity, we extend it at the global statement-level. Some approaches capture the meaning of longer pieces of text by taking the means of the individual term vectors [14], [15]. However, means or sums are rather poor ways of describing the distribution of word embeddings across a semantic space. It would be desirable to capture more properties of the two texts, especially with respect to the semantics of words that do or do not match.

We overcome the above-mentioned limitations by deriving the statement-level similarity from the word-level similarity based on two characteristics: the distribution of words in each requirement statement; and the specificity of each word in the requirements document. To do that, we got inspiration from the work of Mihalcea et al. [12], to derive the statement-level semantic similarity from the word-level semantic similarity. We used hence the Mihalcea’s scoring function for text similarity computation to compute the similarity of each pair of requirement statement (see Equation 2).

First, we identify for each word \( w \) in the text requirement \( R_t \), the word \( w \) in the text requirement \( R_s \) that have the highest semantic similarity \( \text{maxSim}(w, R_t) \) (Equation 1), based on the word-to-word semantic similarity \( \text{wordSim}(w, w) \) using word2vec. Next, the same process is applied to determine the most similar word in \( R_t \) starting with words in \( R_s \).

\[
\text{maxSim}(w_1, R_s) = \max_{w_2 \in R_s} \text{wordSim}(w_1, w_2) \tag{1}
\]

The word similarities are then weighted with the corresponding word specificity using the Inverse Document Frequency (idf) weighting technique to capture the specificity of a word. In a nutshell, this technique aims to measure how much a word contributes to the relevance of two texts. The weighted word similarities are then summed up and normalized with the length of each text segment. The resulting similarity scores are combined using a simple average and thus, the semantic similarity of two requirements \( R_t \) and \( R_s \) is computed as follows:

\[
\text{sim}(R_t, R_s) = \frac{1}{2} \times \frac{\sum_{w \in R_t} \text{maxSim}(w, R_s) \times \text{idf}(w)}{\sum_{w \in R_t} \text{idf}(w)} + \frac{\sum_{w \in R_s} \text{maxSim}(w, R_t) \times \text{idf}(w)}{\sum_{w \in R_s} \text{idf}(w)} \tag{2}
\]

Ultimately, by applying the equation (2), we obtain the final similarity matrix of each pair of requirements.

C. Clustering:

Textual requirements clustering refers to the process of taking a set of requirements and grouping them based on a similarity measure so that, requirements in the same cluster are similar and requirements in different clusters are different. In this context, we adopt the clustering of functional requirements based on their semantic similarity.

Clustering methods can be classified either as hierarchical or partitional [16]. Partitional clustering algorithms such as k-means, require the number of clusters. Thus, they rely heavily on the analyst’s knowledge, as they require the identification of the number of clusters to be generated in advance. In order to reduce the manual intervention, we employ the Hierarchical Agglomerative Clustering algorithm (HAC) [17] as it does not require us to pre-specify the number of clusters in advance. Hence, we utilize the similarity values for each pair of requirements as clustering criterion, taking the semantic similarity matrix of the functional requirements as input for HAC. The HAC algorithm works in a bottom-up manner, each requirement statement is initially considered as a single-element
cluster (leaf). At each step of the algorithm, the two clusters that are the most similar are combined into a new bigger cluster (node). This procedure is iterated until all requirements are member of just one single big cluster, resulting in a hierarchical clustering tree.

However, identifying the optimal number of clusters is not a trivial task. It might be subjective as it can heavily rely on the analyst’s knowledge. In order to automate this task, we implement an operation that identifies automatically the best number of clusters using the Dunn index [18]. The Dunn index is an internal validity index used to evaluate the clustering result when the number of clusters is unknown. Hence, in order to achieve an optimal number of clusters, we calculate the Dunn index each time, when varying the number of clusters. A higher Dunn index indicates better clustering solution. Consequently, to estimate optimal number of clusters that are generated by HAC, we select the number of clusters for which we have a higher Dunn index.

IV. EXPERIMENTAL EVALUATION SETTINGS

In order to assess our approach, we report in this section the research questions that were investigated as well as the four case studies we did.

A. Research questions:

As our study focuses on the automatic grouping of the functional software requirements into a set of clusters, we investigated the following research questions to evaluate the approach:

- **RQ1: To what extent is the proposed clustering solution accurate?**

**Motivation.** For this research question, we aim at determining the accuracy of the proposed clustering solution in order to assess whether our approach succeeded to identify semantic clusters that reflect the domain functionalities embedded in a given functional requirements document.

**Approach.** To answer this research question, we evaluate the proposed clustering solution using two validation criteria as follows:

- **The correctness of the identified semantic clusters:**

This validation criterion aims at verifying whether the identified semantic clusters are close to the semantic clusters provided in the software requirements specification (SRSs) documents. For this, we rely on two well-known measures in the Information Retrieval (IR) field. These metrics are precision and recall [19]. The identified clusters are compared with the reference clusters provided in the SRSs documents, which serve as a ground truth for our evaluation.

Let True Positive (TP) elements be the similar requirements correctly assigned to the same cluster, False Positive (FP) elements be dissimilar requirements assigned to the same cluster and False Negative (FN) elements be similar requirements incorrectly assigned to different clusters. The evaluation metrics are defined as follows:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]

\[
\text{Recall} = \frac{TP}{TP + FN}
\]

- **The clustering gap (C\_Gap):**

For this validation criterion, we aim to verify whether the identified number of clusters is close to the reference number provided in the SRSs document. This is recognized as the clustering gap (C\_Gap). The C\_Gap compares the identified number of clusters with the reference number of clusters. Thus, it is defined as follows:

\[
C\_Gap = \text{number}_{\text{identified clusters}} - \text{number}_{\text{reference clusters}}
\]

- **RQ2: Is the proposed clustering solution practical in realistic settings?**

**Motivation.** For this research question, we aim to establish whether our approach is scalable. Particularly, the goal is to check how well the clustering solution performs when increasing the number of functional requirements.

**Approach.** In order to solve this research question, we assess the following validation criterion:

- **The end-to-end execution time of the clustering solution:**

It consists in measuring the impact of the number of software requirements for each case study on the execution time. Hence, this validation criterion aims to check whether the proposed clustering solution runs within reasonable time for larger number of functional requirements in realistic settings.

B. Case studies:

We assess the applicability of our approach using the software requirements specification documents of four open-access projects from different domains and with different sizes: the E-Store software consists of online sales, distribution and marketing of electronics [20]. The WASP system is a public, real-world requirements specification of context-aware mobile telecommunication services [21]. The UUIS system - Unified University Inventory System - is used to integrate three faculties’ databases providing a web interface that allows user to access and manage the integrated inventory [20]. The MHC-PM system is a Mental Health Care Patient Management System [22].

The table below shows the characteristics of each case study in terms of number of requirements as well as the number of clusters in each SRSs document.

<table>
<thead>
<tr>
<th>CASE STUDY</th>
<th>NUMBER OF REQUIREMENTS</th>
<th>NUMBER OF CLUSTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-Store system</td>
<td>62</td>
<td>20</td>
</tr>
<tr>
<td>WASP system</td>
<td>66</td>
<td>14</td>
</tr>
<tr>
<td>UUIS system</td>
<td>25</td>
<td>11</td>
</tr>
<tr>
<td>MHC-PM system</td>
<td>19</td>
<td>6</td>
</tr>
</tbody>
</table>

V. RESULTS ANALYSIS

In this section, we evaluate the results of applying our proposal to the four aforementioned case studies through the two previous described RQs.
A. Answering RQ1: To what extent is the number of identified clusters correct?

- The correctness of the identified semantic clusters:

In Tables II, we present an example of the reference and the identified cluster for the E-Store system. The requirement statement shown in bold is the identified cluster and an irrelevant functional requirement in that cluster.

In order to answer RQ1, we evaluate our clustering results in terms of precision and recall. We also compare our results to the work in [10]. In fact, the approach used in [10] closely relates to our work as it proposes a method to semantically cluster functional requirements. Thus, we use the work in [10] as a baseline. Table III shows the evaluation results of our approach as well as the baseline [10] in terms of precision and recall. The best results are in bold.

### Table I. Example of identified and reference cluster for the E-Store system

<table>
<thead>
<tr>
<th>Identified cluster</th>
<th>Reference cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system shall allow user to create profile and set his credential.</td>
<td>The system shall allow user to create profile and set his credential.</td>
</tr>
<tr>
<td>The system shall authenticate user credentials to view the profile.</td>
<td>The system shall authenticate user credentials to view the profile.</td>
</tr>
<tr>
<td>The system shall allow user to update the profile information.</td>
<td>The system shall allow user to update the profile information.</td>
</tr>
<tr>
<td>The system shall allow user to register for newsletters and surveys in the profile.</td>
<td></td>
</tr>
</tbody>
</table>

### Table II. Example of identified and reference cluster for the E-Store system

<table>
<thead>
<tr>
<th>Identified cluster</th>
<th>Reference cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system shall allow user to create profile and set his credential.</td>
<td>The system shall allow user to create profile and set his credential.</td>
</tr>
<tr>
<td>The system shall authenticate user credentials to view the profile.</td>
<td>The system shall authenticate user credentials to view the profile.</td>
</tr>
<tr>
<td>The system shall allow user to update the profile information.</td>
<td>The system shall allow user to update the profile information.</td>
</tr>
<tr>
<td>The system shall allow user to register for newsletters and surveys in the profile.</td>
<td></td>
</tr>
</tbody>
</table>

Precision values take a high-range (0.74 – 0.87) and recall values take a reasonable range (0.63 – 0.75) across different case studies. In most of these case studies, we achieved better precision and recall values compared with the baseline [10]. For example, for the MHC-PM case study our approach achieves better precision and recall values by 6 and 16 percentage points respectively. Thus, the evaluation shows clustering results with relatively high quality with better precision and recall values in most case studies compared with the baseline [10].

### Table III. Precision, Recall and C_Gap Values for each Case Study

<table>
<thead>
<tr>
<th></th>
<th>E-store</th>
<th>WASP</th>
<th>UUIS</th>
<th>MHC-PM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Our approach</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.83</td>
<td>0.87</td>
<td>0.74</td>
<td>0.84</td>
</tr>
<tr>
<td>Recall</td>
<td>0.68</td>
<td>0.63</td>
<td>0.75</td>
<td>0.73</td>
</tr>
<tr>
<td>C_Gap</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>The baseline</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.80</td>
<td>0.83</td>
<td>0.72</td>
<td>0.78</td>
</tr>
<tr>
<td>Recall</td>
<td>0.61</td>
<td>0.54</td>
<td>0.60</td>
<td>0.57</td>
</tr>
<tr>
<td>C_Gap</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

At the light of these results, the answer to the first research question (RQ1) is that our clustering solution succeeded to achieve relatively accurate results that can be applicable.

- The clustering gap (C_GAP):

TABLE III shows, for each case study the clustering gap between the identified and the reference clusters. By comparing these results with TABLE I, we note that the identified number of clusters is the same or very close to the reference number of clusters. Moreover, by comparing with the baseline [10], our method succeeded to identify a number of clusters that is closer to the reference number of clusters for the three case studies: E-store system, WASP system and UUIS system.

Therefore, the answer to this research question is that the identified number of clusters is very similar to their corresponding reference number of clusters. In summary, we conclude that the number of the identified cluster is accurate and achieves better results than the baseline [10] in most case studies.

B. Answering RQ2: Is the proposed clustering solution practical in realistic settings?

In order to answer this research question, we measure the execution time of our approach for the four case studies on a laptop with a 2.10 Ghz Intel (R) Core (TM) i7-4600U CPU and a 8GB of memory. In Table IV, we measure the impact of the number of requirements on the end-to-end execution time in order to assess the applicability of our solution.

### Table IV. Execution Time by Number of Requirements

<table>
<thead>
<tr>
<th></th>
<th>MHC-PM</th>
<th>UUIS</th>
<th>E-Store</th>
<th>WASP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of requirements</strong></td>
<td>19</td>
<td>25</td>
<td>62</td>
<td>66</td>
</tr>
<tr>
<td><strong>Execution time in seconds</strong></td>
<td>20</td>
<td>24</td>
<td>41</td>
<td>52</td>
</tr>
</tbody>
</table>

Table IV shows that our approach runs in few seconds for the four case studies. Moreover, Figure 2 shows a linear growth trend for the impact of the number of requirements on the execution time. Given such linear relation and the fact that the end-to-end execution time takes few seconds, the answer to RQ2 is that our approach runs in reasonable time.

In summary, we anticipate that our clustering solution should be practical for much requirements documents.
VI. THREATS TO VALIDITY

In this section, we discuss the limitations of our proposal in terms of internal threats, construct threats and conclusion threats. These threats are as follows:

Internal validity. With regard to computing word similarity, some domain-specific words do not occur in the corpus used to train the word vector space, which might slightly affect the efficiency of word similarity computation. To mitigate this limitation, we map such words to a random vector.

External validity. Our approach is capable of generating clusters from short text requirements. However, if a requirement describing a functionality is of too many sentences, our approach may not provide an accurate result. Hence, in this case, some manual semantic analyses may still be needed to overcome this limitation.

Conclusion validity. We evaluate the applicability of our approach on four open-access projects. Although the evaluation results are promising, the results from just four domains may be not enough to support the conclusion. Thus, we need to evaluate the approach on larger number of case studies for a better evaluation.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented an approach to automatically group functional requirements into semantic clusters in order to breakdown automatically a system into sub-systems at early stages, providing to the system architect a first high-level architecture description of her/his system.

The core of the approach is a clustering solution that is based on the semantic similarity of the natural language requirements. In order to improve the accuracy of the clustering solution, semantic information of both the words and requirements is analyzed and used for compute the similarity. Word-level similarity was firstly computed using word2vec as pre-trained predictive model then, it was extended to the requirements-level using the Mihalcea scoring formula for text similarity computation. Then, we employ the HAC algorithm to cluster functional requirements into semantic clusters. Moreover, we propose and automatic identification of the optimal number of clusters in order to reduce the manual intervention.

To assess the applicability of our approach, we conduct four case studies from open-access projects from different domains and evaluate the results in terms of precision, recall and execution time. Evaluation results reveal that we succeeded to achieve relatively accurate semantic clusters fully automatically within a practical execution time that takes few seconds. Moreover, a comparison with a related work shows that our approach provides relatively better clustering results.

As future work, we will focus on extending our clustering solution to categorize non-functional requirements according to their type. Indeed, most of the work focusing on automating non-functional requirements categorization use supervised learning techniques requiring huge training datasets, which are not always available for all domains. So far, employing clustering to categorize non-functional requirements did not provide sufficient accuracy. Hence, we plan to integrate other techniques taking into account popular key words of each non-functional requirement type to enhance the categorization process.

REFERENCES

Investigating Process Algebra Models to Represent Structured Requirements for Time-sensitive CPS

Mathilde Arnaud * Boutheina Bannour * Arnault Lapitre * Guillaume Giraud *
* Université Paris-Saclay, CEA, LIST
* PES R&D Department, RTE

Abstract

Cyber-Physical Systems (CPS) contain complex computational components that control physical entities. The design of these components must take into account the real-time and concurrent nature of these systems. Formulating requirements that describe CPS behaviors precisely, ruling out misunderstandings, is a crucial yet difficult endeavor.

To increase trust in the requirements, formal methods can be used to check relevant properties of the requirements. We investigate a process algebra to capture real-time behaviors and concurrency in CPS requirements in order to automate their analysis. We use a structured natural language to first express CPS requirements: this takes into account current practice, indeed requirements should be easily writable as well as graspable by stakeholders with various points of view and ease communication among them. At the same time, requirements analysis using simulation or formal validation is possible by taking advantage of the requirements structure. We discuss translation from the structured requirements into the process algebra to automate the overall process. Our approach is implemented and is illustrated by an example issued from CPS4EU project.

1 Introduction

Early validation of Cyber-Physical Systems (CPS) requires the consolidation of requirements, a tedious task due to the nature of CPS behavior. Indeed, physical devices have to be reactive, available and resilient within acceptable time frames, and their control logic can be quite complex. Cross-checking CPS industrial requirements, still mostly expressed in natural language, presents a major challenge, as missing or contradictory requirements can create a costly misunderstanding in the CPS development process. Formal methods can help meet this challenge by validating CPS requirements. In order to apply automated formal analyses, requirements need to be specified in a precise and formal way, through the use of patterns for instance. Fill-in templates facilitate clearer specification of event-driven, state-driven system behaviors. The approach described in this paper specifies CPS requirements following EARS [13] templates, in accordance with recommendations from the International Council on Systems Engineering. Real-time details are introduced to refine event-driven, state-driven system behaviors. An transformation of such requirements into a Process Algebra (PA) is proposed. This process algebra has been implemented in the model-based symbolic execution tool DIVERSITY [10]. Comparatively to other PA available tools, we support real-time behavior modeling. Likewise, we go a step further by considering real-time aspects compared to related works on formalizing requirement using PA until recently [6, 1, 14]. The implementation of PA benefits from existing sat-based techniques integrated in DIVERSITY to support models of timed symbolic automata [3]. Via the transformation, behaviors of CPS systems specified by the requirements can be explored in the tool. Edition and transformation of the requirements are prototyped as a web application using Jupyter Notebook environment. A simplified version of a CPS4EU case study illustrates the proposed approach throughout the paper.

2 Requirement specification

Illustrative example. We illustrate our approach by showing how it can be applied on a real-world use case coming from CPS4EU: electrical networks involving intermittent energy sources. To avoid overload without raising the overall network capacity, it is necessary to manage dynamically the flow of electricity through levers such as batteries or production modulation. Which mechanisms to trigger must be determined very quickly and this role must therefore be entrusted to a software component called NAZA.

We used our approach to analyze NAZA functional requirements. For the purpose of illustration, we choose a subset of requirements: R1 to R6A in Tab.1, written in Structured

---

1 This work was financially supported by European commission through CPS4EU project that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826276. The JU receives support from the European Union’s Horizon 2020 research and innovation programme and France, Spain, Hungary, Italy, Germany.

DOI reference number: 10.18293/SEKE2021-147

---

2 Some references and tools process algebra are CADP/LOTOS (https://cadp.inria.fr/) and FDR/CSP (https://cocotec.io/fdr/).
Natural Language as presented below. The NAZA automaton is in charge of computing levers setpoints (cf R1). When the computation is successful, it then uses consensus (cf R2) and sends the results to middleware (cf R3, R4, R5). When the computation fails, it must launch a backup logigram algorithm (cf R6A). Our analysis revealed possible deadlocks and we proposed to replace requirement R6A by requirement R6B if no result upon levers setpoints calculation (R1), the NAZA Core shall execute backup algorithm within [10, 60] seconds.

Table 1: Excerpt requirements on levers setpoints calculation in NAZA (Nouveaux Automates de Zones Adaptatifs)

3 Target process algebra

Time datatype, actions and modes. Clocks are typed in a dense time domain $T$ isomorphic to the set of positive rational numbers $\mathbb{Q}_+$. Given a set of clocks $\text{Clk}$, a clock valuation $\nu$ is a mapping $\nu : \text{Clk} \rightarrow T$. The set $\mathcal{F}(\text{Clk})$ of clock formulas is built up recursively out of logical conjunction and atomic formulas of the form $\text{True}$, $\text{False}$, $\text{clk} \equiv d$, where $d$ is a constant duration (typed in $T$) and $\equiv \in \{<, \leq, >, \geq\}$. The set of clock invariants $\mathcal{I}(\text{Clk})$ is defined by conjunctions of formulas of the form $\text{clk} \equiv d$, where $\equiv \in \{<, \leq\}$. Let $\text{Act}$ be a set of actions which contains the silent action $\tau \in \text{Act}$, and $A \subseteq \text{Act} \setminus \{\tau\}$ be a partition $I \cup O$. Elements $a$ of $I$ (resp. of $O$) are called inputs and denoted by $?a$ (resp. called outputs and denoted by $a$). In a parallel composition, inputs and outputs can synchronize resulting in $\tau$. We denote $?a = a$ (and vice versa). Let $M$ be a set of modes with initial mode $m_0 \in M$.

Processes. A process is defined by the following syntax:

$$ P :::= \sum_{i \in I} \alpha_i.P_i \mid \text{inv}(\psi).P \mid \text{nil} \mid P_1|P_2 \mid K $$

$\alpha ::= (m, \phi, a, R, m')$ are the basic building blocks of the syntax. They are described by an enabling mode $m \in M$, an enabling clock formula $\phi \in \mathcal{F}(\text{Clk})$, an action $a \in \text{Act}$, a set of clocks $R \subseteq \text{Clk}$ to be reset, and a target mode $m' \in M$ to evolve into. Some of these elements can be dropped. For instance $(\phi, a, R)$ denotes that enabling mode can match any arbitrary mode and that no mode change is to be made. The construct $\text{inv}(\psi).P$ defines a clock invariant $\psi \in \mathcal{I}(\text{Clk})$ that has to be satisfied on time passing for the execution of the process $P$. This notion is borrowed from timed (and hybrid) automata and requires some technical handling at the evaluation of the process. All other constructs are classic [5]: the empty process $\text{nil}$, action prefixing $\alpha_i.P_i$, nondeterministic choice $\sum_{i \in I} \alpha_i.P_i$, parallel composition $P_1|P_2$ with possible synchronization of input / output actions, the process constant $K \mathrel{def} P$ for recursive definition.

Process execution. The key idea is to ensure that time
progress cannot invalidate either of the local invariants of parallel processes. We introduce a construct of global invariants at evaluation that will be updated upon the evaluation of the left or right processes in a parallel composition. A global invariant or g-invariant is defined by the syntax: 
\[ \Psi := \psi \mid \Psi \land \Psi \mid \Psi_{L} \mid \Psi_{R} \mid \Psi_{L,R} \psi, \text{ with } \psi \in \mathcal{I}(\text{Clk}). \]
We define functions \( L, R \) and \( f \) on g-invariants that return respectively the left side of the g-invariant, the right side, and the formula denoting conditions at the time of evaluation. If \( \Psi \) is in a decomposed form \( \Psi_{1} \land X \Psi_{2} \) with \( X \in \{\_L, R\} \) then \( L(\Psi), R(\Psi) \) and \( f(\Psi) \) denote \( \Psi_{1}, \Psi_{2} \) and \( f(\Psi_{1}) \land f(\Psi_{2}) \) respectively, otherwise \( L(\psi), R(\psi) \) and \( f(\psi) \) are \( \psi \). The process execution is defined up to an execution context \( ec = (m, v, \Psi) \) which represents the necessary information to perform an execution step, namely the current mode \( m \), the current valuation \( v \) of clocks and the current g-invariant \( \Psi \) to be applied. Operational rules of the execution are defined as follows:

**Rule ATOM**

\[ \vdash (m, \phi, a, R, m'). P \overset{a}{\rightarrow} (m', \phi), \Psi) \]

with \( \psi_{L} = v_{L} \text{clk} \rightarrow \text{clk+1}, \text{clk} \in \text{Clk}, d \in T, v_{0} = \phi \land f(\Psi) \) and \( \psi_{R} = v_{R} \text{clk} \rightarrow 0, \text{clk} \in \text{R} \).

**Rule INV**

\[ \vdash \text{inv}(\psi). P \overset{a}{\rightarrow} \text{P}(m, v, \Psi) \]

with \( v = f(\Psi) \), and \( \psi_{f} = \text{ginnv upd}(\Psi, \psi) \).

**Rule SUM**

\[ P \overset{a}{\rightarrow} \text{P}_{1}(m, v, \Psi) \overset{a}{\rightarrow} \sum_{i \in I} \text{P}_{i}(m', v', \Psi') \]

**Rule CONST**

\[ P \overset{a}{\rightarrow} \text{P}' \overset{a}{\rightarrow} K \overset{a}{\rightarrow} \text{P}' \overset{a}{\rightarrow} \text{P}' \overset{a}{\rightarrow} \text{P}' \]

with \( K \overset{a}{=} P \).

**Rule PAR1L**

\[ P_{1} \overset{a}{\rightarrow} (m, v, \text{L}(\Psi) \land \text{ginnv}(P_{2}, R(\Psi))) \overset{a}{\rightarrow} \text{P}'_{1}(m', v', \Psi') \]

\[ \vdash P_{1}[P_{2} \overset{a}{\rightarrow} (m, v, \Psi)] \overset{a}{\rightarrow} \text{P}'_{1}[P_{2} \overset{a}{\rightarrow} (m', v', \Psi')] \]

with \( \Psi' = \text{ginnv upd}(\text{L}(\Psi) \land \text{R}(\Psi), \Psi_{1}') \)

**Rule PAR2L**

\[ P_{1} \overset{a}{\rightarrow} (m, v, \text{L}(\Psi) \land \text{ginnv}(P_{2}, R(\Psi))) \overset{a}{\rightarrow} \text{P}'_{1}(m', v', \Psi') \]

\[ P_{2} \overset{a}{\rightarrow} (m, v, \text{ginnv}(P_{1}, \text{L}(\Psi))) \overset{a}{\rightarrow} \text{P}'_{2}(m', v', \Psi') \]

\[ \vdash P_{1}[P_{2} \overset{a}{\rightarrow} (m, v, \Psi)] \overset{a}{\rightarrow} \text{P}'_{1}[P_{2} \overset{a}{\rightarrow} (m', v', \Psi')] \]

with \( v'(\text{clk}) = 0 \) if \( v'_1(\text{clk}) = 0 \) \& \( v'_2(\text{clk}) = 0 \),

else \( v'(\text{clk}) = v'_1(\text{clk}) + v'_2(\text{clk}) \), for any clock \( \text{clk} \in \text{Clk} \).

The process execution is inductively defined on the form of the process term. Action prefixing \( (m, \phi, a, R, m'). P \) evolves to \( P \) under the constraint that time elapsing is compatible with its clock formula \( \phi \) and formula of current g-invariant \( f(\Psi) \). In case of invariant definition \( \text{inv}(\psi). P \), then function \( \text{ginnv upd} \) is called to update the relevant side of the g-invariant for next action executions in \( P \) and other parallel processes if any. In fact, the case of parallel composition is the most subtle, other rules are classic [5]. It uses a function \( \text{ginnv} \) which computes the formula of a g-invariant of process \( P \) given a current g-invariant \( \Psi \) (of an \( ec \)). Concerned part of \( \Psi \) is returned in case no new invariant is encountered. For clarity’s sake, functions \( \text{ginnv upd} \) and \( \text{ginnv} \) are defined using the Pattern-Matching notation of OCaml 3 as follows:

\[ \text{ginnv upd}(\Psi, \Psi') = \text{match } \Psi \text{ with} \]

\[ | \psi \rightarrow \Psi' | \Psi_{1} \land \Psi_{2} \rightarrow \Psi' | \Psi_{1} \land R \Psi_{2} \rightarrow \Psi_{1} \land R(\Psi') \]

\[ \text{ginnv}(P, \Psi) = \text{match } P \text{ with } \]

\[ | \sum_{i \in I} \alpha_i.P_i \rightarrow f(\Psi) | \text{nil} \rightarrow f(\Psi) \]

\[ | \text{inv}(\psi). P \rightarrow \psi \]

\[ | P_{1}[P_{2} \rightarrow \text{ginnv}(P_{1}, \text{L}(\Psi) \land \text{ginnv}(P_{2}, R(\Psi))) \]

\[ | K \rightarrow \text{ginnv}(P) \text{ if } K \overset{a}{=} P \]

An illustrative execution of two parallel processes is given in Fig.1. It shows how actions \( a \) and \( b \) will be interleaved in the presence of invariants. From context \( ec_{0} = (m_{0}, d, \text{clk} \rightarrow 0), \Psi_{0} = \text{True} \), both left process \( P_{1} \) and right process \( P_{2} \) are evaluated. \( P_{2} \) can evolve into \( P_{2}' \) which allows to reach context \( ec_{3} \). \( P_{1} \) cannot be executed, as rule \( PAR1L \) requires \( g \)-invariant \( L(\Psi_{0}) \land \text{ginnv}(P_{2}, R(\Psi_{0})) \) to be satisfied for \( P_{1} \) to be executed, with \( L(\Psi_{0}) = \text{True} \) and \( \text{ginnv}(P_{2}, R(\Psi_{0})) = \text{clk} < 1 \). The execution of \( P_{1} \) is enabled by formula \( \text{clk} = 1 \), which cannot be satisfied at the same time as g-invariant \( \text{clk} = 1 \) at this point of execution. In the evaluation, once the g-invariant \( \text{True}_{\land} \text{clk} = 1 \) is applied on some process (here left), it is returned in a neutral form \( \text{True}_{\land} \text{clk} < 1 \). Later on, from context \( ec_{3} \), the execution of process \( P_{1} \) becomes possible: rule \( PAR1L \) applies as g-invariant becomes \( \text{True}_{\land} \text{True} \text{ (previously } \text{True}_{\land} \text{clk} = 1 \text{ which allows time elapsing with any delay.} \]

\[ \text{Figure 1: Parallel execution} \]

The transformation process described in Section 4 generates the NAZA processes given in Fig.2. We indicate each time the tuple of requirement identifiers that allowed the inference. Fig.4 is a graphical view of the NAZA Core process obtained from requirements \((R1, R2, R3, R4, R5, R6.A)\).

\[ \text{https://ocaml.org/} \]
Core $\overset{def}{=} \text{Core}^\prime$;

$\text{inv}(clk_k \leq 5).c\text{alc\_setpoints}$

.$\{\text{new\_setpoints}.\text{determine\_common\_fevers}.\text{consensus} \}

.$\{\text{send\_batteries\_setpoints}.([clk_k = 5], [clk_k]).\text{Core} \}

.$\{\text{send\_topological\_orders}.([clk_k = 5], [clk_k]).\text{Core} \}

.$\{\text{send\_modulation\_orders}.([clk_k = 5], [clk_k]).\text{Core} \}$

$+\text{no\_result}.([clk_k = 5], [clk_k]).\text{Core}^\prime)$

$(R1, R2, R3, R4, R5, R6A)$

Core$^\prime \overset{def}{=} \text{Core}^\prime$;

$\text{inv}(clk_k \leq 5).c\text{alc\_setpoints}$

.$\{\text{new\_setpoints}.\text{determine\_common\_fevers}.\text{consensus} \}

.$\{\text{send\_batteries\_setpoints}.([clk_k = 5], [clk_k]).\text{Core} \}

.$\{\text{send\_topological\_orders}.([clk_k = 5], [clk_k]).\text{Core} \}

.$\{\text{send\_modulation\_orders}.([clk_k = 5], [clk_k]).\text{Core} \}$

$+\text{no\_result}.([clk_k = 5], [clk_k]).\text{Core}^\prime)$

$(R1, R2, R3, R4, R5, R6B)$

Supervisor $\overset{def}{=} \text{Supervisor}^\prime$

$\{\text{nominal}\}.\text{no\_result}.([clk_k], \text{\_backup}).\text{Supervisor} + \text{[backup]}\text{.inv}(clk_k \leq 60)$

.$\{[10 \leq clk_k \leq 60], \text{execute\_backup\_algorithm}, \text{\_nominal} \}$

$\text{.inv}(\text{True}).\text{Supervisor}$

+$\{[\text{backup}], \text{\_new\_setpoints}, \text{\_nominal} \}.\text{Supervisor}^\prime$

$(R6B, R7, R8)$

**Figure 2: NAZA processes**

**Exploration** For any finite sequence of $P_0 \overset{a_0}{\rightarrow} P_1 \overset{a_1}{\rightarrow} \ldots \overset{a_{n-1}}{\rightarrow} P_n$, we note $P_0 \overset{a_1 \ldots a_n}{\rightarrow} P_n$, or simply $P_0 \overset{\text{new}}{\rightarrow} P_n$. Our operational approach to exploration relies on small steps of the form $P \overset{a}{\rightarrow} P'$ which corresponds to the execution in a process $P$ of an action $a$. This leads to a new process $P'$ that synthesizes possible executions that may occur after action $a$. Potential **deadlocks** can be detected typically for a sequence $P_0 \overset{a}{\rightarrow} P_n$ such that no successor process $P_{n+1}$ can be computed from $P_n$. A deadlock arises in process Core (because of $R6A$). The execution of the backup algorithm in Core is constrained by the formula $10 \leq clk_k \leq 60$ which is not compatible with the formula of g-invariant $clk_k \leq 5 \land clk_k \leq 60$. The ambition comes from missing requirements on some parallel execution which is in charge of the backup algorithm, unconstrained by the period of 5 seconds. It is hereafter specified by a dedicated process Supervisor which executes in parallel with the new Core'.

The full NAZA case study introduces requirements for an additional complex process Acquisition, itself divided into a number of parallel processes with various periods: every 10 seconds, the NAZA Acquisition shall request new datapoints from middleware and transmit them to NAZA Core; if no data is received within 10 minutes upon datapoints request, the NAZA Supervisor shall enter in fault mode; every 1 minutes, the NAZA Acquisition shall request new network situation from upper level and transmit them to NAZA Core; etc. We have also applied exploration with the aim of identifying the so-called zeno executions [16], i.e., executions in timed systems with an unbounded number of actions executed in a bounded length of time. The NAZA requirements should not allow such behaviors, as they are not possible in practice. By applying small-steps $P_0 \overset{a_1 \ldots a_n}{\rightarrow} P_n$, loops ($P_n = P_0$) are checked to be non-zeno, i.e., there exists a clock $clk$ and some $i, j \leq n$ such that $clk$ is reset in step $i$ and $clk$ is bounded from below $\epsilon < clk$ in step $j$. Overall, the exploration of processes helped provide a good understanding of the NAZA requirements and assisted their refinement.

**4 Transformation**

We outline next the main transformation patterns into process algebra. System responses sharing the same trigger are composed in parallel (Fig.3a); and triggers can be non-
deterministically produced upon a system response (Fig.3b). Each (sub-)system is assumed to have an implicit initialization upon which its behavior occurs. Repetitive behavior of sub-system is also assumed with a topmost recursion (Fig.3c). It can be associated a period $d$ : time is constrained during the iteration through an invariant of the form $\text{inv}(\text{clk}_1 \leq d)$, and at the end system clock $\text{clk}_1$ is reset through ($[\text{clk}_1 = d], \{\text{clk}_1\}$). A synchronization together with corresponding input / output actions is inferred if a sub-system response is triggered by some other sub-system behavior (Fig.3e). When a system response occurs within an interval $[d_1, d_2]$, a dedicated clock $\text{clk}_2$ is used to encode such constraint ($d_1 \leq \text{clk}_2 \leq d_2$) associated with an invariant (re-)definition $\text{inv}(\text{clk}_1 \leq d \land \text{clk}_2 \leq d_2)$, which sets an upper bound $d_2$ on time elapsing before the response occurs (Fig.3d). State-driven triggers/responses are transformed by various patterns involving enabling modes or mode change constructs of the process algebra.

5 Related Work

Requirements are used as a tool to ensure sound communication between stakeholders for the successful design of the system [15]. Thus, specifying good requirements is important to develop qualitative products that can satisfy user’s needs [4]. Cross-checking CPS industrial requirements, mostly expressed in natural language, presents a major challenge, that formal methods can help meet by validating CPS requirements.

Analyzing requirements written as natural language and getting guarantees of non-ambiguity, completeness, consistency would be ideal. Unfortunately, obtaining formal guarantees from natural language requirements is difficult, as they often lack a structure that would allow to apply formal methods. Some works explore nonetheless the possibilities for extracting useful information from natural language documents, for instance [11] automates a process based on use case documents. The authors use dependency parsing techniques to automatically generate activity diagrams describing use case flow. The input writing style for use case is fixed, so use cases may have to be rewritten according to this style, and some other steps in the method may require manual intervention. The authors provide algorithms to automatically check some structural defects they identify, but results include false positives and false negatives due to the intrinsic ambiguity of natural language. For stronger assurances, other works focus on what formal methods offer.
Several formal methods to validate requirements can be used. We focus on those that take as input natural language requirements or that address real-time, concurrent systems as our goal is to validate CPS requirements. The research problem addressed in [7] is the automatic generation of timed state-rich formal models from natural-language specifications to support test generation. Requirements are expressed according to a controlled language called SysReqCNL, where requirements have the form of action statements guarded by conditions. The approach presented in [9] aims to check some properties or real-time requirements: rt-consistency, consistency and vacuity. Requirements must be written following a tightly constrained English grammar closely related to LTL with the underlying Duration Calculus semantics. Authors of [14] present a formalisation of requirements into a process algebra supported by the tool FDR, used to generate test cases. This work uses a strongly controlled language as input, and requirements may contain data but not time.

Filling the gap between natural language and formal methods is our goal in this paper. In this perspective, a trade-off must be found between, on the one hand, possibly difficult to master, strongly constrained requirements [17], and on the other hand, natural language requirements, that must be transformed into formal languages [2] and may lose the intended meaning in the process [8]. Fill-in templates facilitate clearer specification of event-driven, state-driven system behaviors. Automatically writing and analyzing such semi-formal requirements is still a challenge, especially when taking time into account. Typically, [12] synthesizes from EARS the logic of CPS controllers, however timing details are not formally analyzed.

6 Conclusion

This paper investigate a real-time process algebra to represent structured natural language requirements. Compositional modeling using process algebra provides powerful constructs to build larger processes from smaller ones specified by the unitary structured requirements through transformation techniques. The proposed approach is implemented, which enables to explore the resulting model and thus to better understand the real-time behaviors and concurrency implied by the requirements. These first results have to be consolidated on larger experiments. A possible continuation of this work is to develop refinement or bisimulation methods for the process algebra in order to assist requirements evolution and clarification.

References

Risk Analysis for Collaborative Systems during Requirements Engineering

Kirthy Kolluri, Robert Ahn, Lawrence Chung
Department of Computer Science
The University of Texas at Dallas
Richardson, TX, USA
{kirthy.kolluri, robert.sungsoo.ahn, chung}@utdallas.edu

Tom Hill
Fellows Consulting Group
Dallas, TX, USA
tom@fellowsconsultinggroup.com

Abstract— Risk, a potential occurrence of some undesirable event, can be dangerous if not adequately identified and dealt with early on during software development. However, identifying risks can be difficult, hence oftentimes resulting in a particular software system that is unable to address risks, especially critical ones adequately. This paper proposes an ontology-based framework for performing risk analysis with the Augmented Reference Model - The Reference Model augmented with risk analysis. The Reference Model emphasizes that the user requirements are met through the collaboration between the system and the events occurring in its environment - i.e., not by the system alone, hence the term "collaborative system." We also offer an activity-oriented ontology to carry out risk analysis by identifying risks from negating the events in the environment and system. Such negations of the requirements, specifications, and domain events generate a graph-like representation, called Risk Analysis Graph (RAG), to help perform risk analysis. To validate our framework, we have performed two experiments using questionnaires to identify risks and use the risk analysis tool to generate RAG for performing risk analysis. We feel that at least these experiments show that RAG helps identify risks - especially the critical and uncommon ones that we would not have thought of.

Keywords— Risk; Risk Analysis; Ontology; The Reference Model (WRSPM Model); Requirements Engineering

I. INTRODUCTION

Risk, which is defined as a situation or event where something of human value (including humans themselves) has been put at stake and where the outcome is uncertain"[10], is a phenomenon faced or caused by the agent (e.g., User, Software or Hardware). If the requirements do not address critical risks as fundamental potential problems, the projected system may lead to grave consequences [1]. For instance, in building a smartphone app for helping blind people navigate indoors, it might not be too evident to requirements engineers that a blind person may not be able to walk straight in line or figure out where to turn. This is just one example of, among many such potential risks.

Risks involving the user and the system may arise due to the system malfunctioning or the user misusing the system. For example, a blind person has to walk ten steps before making a right turn. What if the smartphone application asks the blind person to turn earlier or later after walking ten steps? Or what if the user ignores the instructions and fails to turn at the right spot? Addressing these kinds of scenarios by the requirements engineers and the software developers before developing the application would help plan with risk minimization and mitigation strategies.

The Reference Model (WRSPM Model) [3] emphasizes that the user requirements are satisfied by the collaboration between the user and the events in its environment. Since it involves both the system and the user, the term collaborative system is used (e.g., a smartphone app, Theia1 for helping blind people navigate inside one of our campus buildings ). Keeping Murphy's Law in mind which states, anything that can go wrong will [2], we perform risk analysis by extending the Reference Model that we adopted into the Augmented Reference Model.

Negating the events in the Reference model gives us the possible negative things (risks) that may arise in a particular environment. Using these possibilities, a graph-like structure called the Risk Analysis Graph (RAG) is generated. We use a highly activity-oriented ontology to identify the most important/critical risks obtained by the RAG in performing risk analysis. We have carried out experimentation in two parts and compared the total number of risks obtained/ignored by the students who performed both these experiments. Through this experimentation, we have observed that simple yet important risks, such as walking in a straight line, etc., can be overlooked.

DOI reference number: 10.18293/SEKE2021-192

1 Theia is the Greek goddess of sight
Running example: An indoor navigation app (Theia) for helping blind people is used as the running example to illustrate the fundamental concepts of the risk analysis framework. For ease of understanding, we use the example of a blind person (Stevie) navigating indoors using the smartphone application (Theia). Stevie is a blind person (student) who wants to navigate in the campus building. He uses the smartphone application, Theia, to navigate from his current location.

Section II describes the related work. Section III describes the proposed approach for performing risk analysis. Section IV describes the experimentation and the observations of the experimentation. Section V includes the overall observation and threats to validity. In the end, a summary of the paper is described, along with some future work in Section VI.

II. RELATED WORK

The Reference model draws attention to the vital concept of satisfying the user’s requirements through the collaboration between the environment and the system through events. The environment comprises everything associated with the users (designators), the activities performed by the designators, surrounding infrastructure (e.g., buildings, things, etc.), and the environment events (e) are those that are associated with the environment. The system comprises the software system, the actions performed by the software system, and the programming concepts related to the software system. The system events (s) are those associated with the system. These events are classified as visible and hidden events – i.e., events visible and hidden to the environment and the system - (e_h, e_v) and (s_h, s_v) respectively [3, 4]. These environment events and the system events help satisfy the requirements.

In the area of Requirements Engineering, the Reference Model [3, 4] emphasizes collaboration and focuses on applying formal methods to the user requirements and reducing them to the system specification. We adopt and extend the Reference Model into the Augmented Reference Model to perform risk analysis in this work.

In the area of Risk Analysis, the work discussed in [1] proposes a Goal-Risk (GR) framework for modeling risks during the requirements engineering phase. They model goals, events, and treatments in three layers. The work discussed in [11] builds upon the framework proposed in [1] and provides multi-object optimization; hence more queries related to risk. Some similarities between our work and the work addressed in [1] are the risk analysis performed in the requirements engineering phase and an ontology provided, which analyzes risks. Our framework uses the Augmented Reference Model to perform risk analysis by negating the events (requirements, specification, and domain). The approach proposed in our paper aims to systematically obtain risks that can and cannot be obtained by logical negation.

CORAS [5] is a risk analysis framework that models, analyzes risks, and handles them. Each risk is analyzed in this framework by asking questions and prioritizing risks. Our framework provides an activity-oriented, risk-oriented ontology that addresses critical risks identified while performing risk analysis using the Reference Model and the Risk Analysis Graph. The work discussed in [6, 7, 8] explains obstacle analysis which explains decomposing the goals. They also provide a set of rules, including negation. There is some similarity in the approach, but we use only functional requirements in our work and use negation for obtaining risks.

The ontology of risk discussed in [9] is regarding its relationship with value, unlike our ontology, which is strongly tied to identifying risks that the agents face. We adopt the ontological components addressed in Requirements Modelling Language (RML) [12] and add another ontological concept, “Risk,” to the existing work to tie the concept of Risk to Action and Agent.

III. A FRAMEWORK FOR PERFORMING RISK ANALYSIS

To help find and analyze risks, the risk analysis framework described in this paper uses an activity-oriented framework. This process transforms the Reference Model into the Augmented Reference Model. The Risk Analysis Graph is generated by using negation which is explained in detail in the following steps. A tool to help generate a Risk Analysis Graph (RAG) was also developed.
A. Step 1: Obtain Overall Ontology:

It is essential to explicitly represent high-level concepts such as Agents, Risks, Actions, Requirements, Specification, Domain in a domain-independent approach to avoid omissions and commissions of risks while transforming the Reference Model into an augmented Reference Model and generating risks. Additionally, some concepts may be incorporated from a domain-dependent ontology as well. All the concepts and the relationships between them can be found in Fig. 1.

This ontology is independent of the domain and can be used for various domains which use any kind of collaborative system. In this step, we want to identify the domain-level concepts involved to help the requirements engineers/developers to generate risks. This is an activity-oriented ontology that addresses risks associated with each activity performed by the Agent. The ontology is also used to identify the most critical risks obtained from the risk analysis outcome after step 5.

B. Step 2: Acquire and Decompose Requirements:

The proposed approach uses the functional requirements R from the Reference Model, represented in the form $i \rightarrow t$. This acquired requirement is AND-decomposed into sub-requirements: $R_{if}$ and $R_{then}$. Requirements are decomposed to broaden the scope of the risk generation. Each of these sub-requirements can be further decomposed if there exists an $i \rightarrow t$ relation.

Instance-level requirements, specification, and domain were used throughout this paper for facilitating simplicity in understanding the risk analysis process.

For instance,

$R$: When Stevie indicates his destination as room 3.415, Theia shall ask Stevie to walk 10 steps forward

is AND-decomposed into

$R_{if}$: Stevie indicates his destination as room 3.415

$R_{then}$: Theia shall ask Stevie to walk 10 steps forward

C. Step 3: Generate Specification and Domain:

Using this proposed approach, it is possible to partially automate the specification and domain using the Ontology-based approach, which is discussed further in Step 3. As shown in the Reference Model, since every requirement has a specification and domain, all sub-requirements have sub-specifications and sub-domains, respectively. The specification and domain can be further decomposed if it is of the form $i \rightarrow t$ or if an (AND) or (OR) or (,) or (.) are present. For instance, after decomposing R into $R_{if}$ and $R_{then}$, we obtain the $S_{if}$, $S_{then}$, $D_{if}$ and $D_{then}$ respectively.

Considering the sub-requirement $R_{then}$ (due to space limitation), we obtain

$D_{then}$: The smartphone’s speaker is switched on and is in working condition

$S_{then}$: If the microphone receives a voice input signal, Theia notifies using the speaker with a voice instruction to walk 10 steps forward

The events associated with $D_{then}$ and $S_{then}$ are checked for further refinements and are decomposed based on the satisfaction of the criteria. Since $S_{then}$ is in the form $i \rightarrow t$, $S_{then}$ is decomposed into $S_{then if}$ and $S_{then then}$.

$S_{then if}$: The microphone receives a voice input signal
The augmentation process is shown in Fig. 2 with a detailed instance-level example.

E. Step 5: Obtain Risks by generating the Risk analysis Graph (RAG)

In this paper, we propose the generation of Risk Analysis Graph (RAG), shown in Fig. 3, by the systematic generation of risks that are hard to find. For this systematic generation of risks, we perform logical negation of AND, OR, Implication, etc., as the starting point. Due to the space limitation, we will present the process of obtaining risks by negating the logical implication \(i \rightarrow t\) and another particular case \(\neg (i, A)\). For this work, we have implications in the requirements, specification, and domain as well. We have worked on all possible combinations to obtain various risks from a set of requirements. However, we will be illustrating only the implications associated with the R in the running example due to space limitation. The requirement is of the form \(i \rightarrow t\). \(i \rightarrow t\) can also be written as \(\neg i \lor t\). For instance, P1: \(i \rightarrow t\) can be written as

}\begin{align*}
P1: & \neg (\text{Stevie indicates his destination as room 3.415}) \lor (\text{Theia shall ask Stevie to walk 10 steps forward}) \\
& \text{which is equivalent to} \\
& (\text{Stevie does not indicate his destination as room 3.415}) \lor (\text{Theia shall ask Stevie to walk 10 steps forward}) \\
& \text{Stevie does not indicate his destination as room 3.415 is a risk (when the destination he wants to go to}
\end{align*}

Similarly, for transforming the system events, we substitute the normal and risk cases in equation 3 which is not shown here due to space limitation. The augmentation process is shown in Fig. 2 with a detailed instance-level example.

\[S \text{then}_\text{then}: \text{Theia notifies using the speaker with a voice instruction to walk 10 steps forward}\]

D. Step 4: Perform Augmentation

When considering the Reference Model and transforming the equations from the Reference Model, the phenomenon \(\Phi\) which takes place is a union of the environment events ‘e’ and the system events ‘s’ [3]. Hence,

\[
\Phi = e \cup s \tag{1}
\]

\[
e = e_h \cup e_v, \quad e_h \cap e_v = \Phi \tag{2}
\]

\[
s = s_h \cup s_v, \quad s_h \cap s_v = \Phi \tag{3}
\]

there are four events associated with it (normal case), i.e., \(e_h, e_v, s_h, s_v\) [3, 4]. In this piece of work, we call them normal case events, hence represented by the notation \(e_{\text{normal}}, e_{\text{risk}}, s_{\text{normal}}, s_{\text{risk}}\). Augmenting the Reference Model is about adding risks (negating) to the normal events. The events associated with risks are \(e_{\text{risk}}, e_{\text{risk}}, s_{\text{risk}}, s_{\text{risk}}\). Therefore, in the Augmented Reference Model we have eight events associated with it, both the normal case events and risk events namely. By substituting the normal and risk case events we get,

\[
e_{\text{Aug}} = e_{\text{normal}} \cup e_{\text{risk}} \tag{4}
\]

\[
e_{\text{Aug}} = e_{\text{normal}} \cup e_{\text{risk}} \tag{5}
\]

Similarly, the system events can be obtained as shown in equations 4 and 5. Transformation of the environment and the system events into negated events is done by substituting in equation 2,

\[
e_{\text{Aug}} = (e_{\text{normal}} \cup e_{\text{risk}}) \cup (e_{\text{normal}} \cup e_{\text{risk}}) \tag{6}
\]

\[
(e_{\text{normal}} \cup e_{\text{risk}}) \cap (e_{\text{normal}} \cup e_{\text{risk}}) = \Phi \tag{7}
\]

\[\text{Similarly, for transforming the system events, we substitute the normal and risk cases in equation 3 which is not shown here due to space limitation. The augmentation process is shown in Fig. 2 with a detailed instance-level example.}\]
This risk might have many cases, such as indicating the wrong room number as his destination, not indicating any room number after turning the app on, an unclear indication of his destination, etc. To identify different possibilities of risks, we negate P1 represented as P2.

P2: \( \neg i \rightarrow t \) \( (\text{Stevie does not indicate room 3.415 as his destination}) \lor (\text{Theia shall ask Stevie to walk 10 steps forward}) \)

Negation yields P3, which is

P3: \( (\text{Stevie indicates his destination as room 3.415}) 
\land (\text{Theia shall ask Stevie to walk 10 steps forward}) \)

which would lead to P4

P4: \( (\text{Stevie indicates his destination as room 3.415}) 
\land (\text{Theia shall not ask Stevie to walk 10 steps forward}) \)

which indicates a risk. This case of risk where Theia shall not ask Stevie to walk ten steps forward can be analyzed. Multiple cases could be associated with this risk, such as Theia may ask Stevie to walk eight steps or may ask him to walk 12 steps, etc. How can we try to alleviate this risk? Risk mitigation mechanisms can be designed based on the risks obtained. For instance, to make sure that Stevie walks the correct number of steps, a screen-tapping mechanism can be introduced, where Stevie taps the screen for every step taken to keep a count on the steps taken.

Not all risks can be addressed by logic, and there are some shortcomings as well. Considering the truth values for \( i \rightarrow t \), if ‘i’ is false, irrespective of whether ‘t’ is true or false, the statement \( i \rightarrow t \) is always true [13]. This analysis will help us find a few risks which a simple negation of \( i \rightarrow t \) could not find. For instance, \( t \): \( \text{Theia shall ask Stevie to walk 10 steps forward} \) makes the truth value false, but if \( \text{Stevie does not indicate his destination as room 3.415} \), this entire statement is true according to logic, but in reality, it is not. Similarly, if ‘i’ is false and ‘t’ is false (negated), the entire statement would still be true. Secondly, after analyzing the possibility for risks apart from the logical negation of \( i \rightarrow t \), which is \( \neg (i \lor t) \), it is found that \( (\neg i \land A \land t) \), which cannot be obtained by the logical negation of implication, pulls in risk(s).

Risk Analysis Tool: We developed a risk analysis tool to generate the RAG for performing risk analysis. The formal strategies addressed in step 5, namely \( \neg (i \rightarrow t) \), \( \neg (i \land A \land t) \), \( (i \land A \land t) \), etc. are used as templates for semi-automation of risks using the requirements, specifications, and domains obtained in Step 2, Step 3 and Step 4 and use the ontology captured in Step 1 to identify the most important risks obtained in the semi-automation process. The tool’s images are not shown here due to space limitation, but the results have been discussed briefly.

IV. EXPERIMENTATION

We have experimented in two parts to validate our risk analysis process through 1) group projects of several undergraduate, graduate-level, PhD-level requirements engineering courses, which one of the coauthors has been teaching for more than 12 years 2) generating the RAG to obtain risks and analyze them. Most of the students involved in the experimentation for both parts of the experiments were majoring in Computer Science. Students learned about the concepts related to the Reference Model, Ontology, etc., as a part of their coursework and applied this knowledge along with using Murphy’s Law to perform both parts of the experiment, respectively.

For experiment 1, the problem domain for all the projects was to develop a smartphone app for helping blind people navigate indoors that was not the same. We have selected around 30 projects, with approximately four students on average in each project. They developed a questionnaire for identifying the risks that may arise while the blind person navigates indoors before developing the actual application.

![Figure 4. Graph depicting the risks found using questionnaires vs. RAG](image)

This study has shown that students could find out risks but lacked identifying critical and uncommon risks. The Teaching Assistant (TA), one of this paper’s coauthors, has carried out a detailed review and analyzed different kinds of risks obtained by the teams. The questionnaires were able to identify some risks. Since the questionnaires developed were at a graduate-level, their analysis was restricted to being very basic and shallow. The risks identified by the teams were at a brainstorming-level when compared to the risks identified by developing the RAG.

For the second part of the experiment, 30 PhD. and 30 senior-level graduate students volunteered to help us experiment. Every student was provided with the initial version of the tool required to generate the RAG. The students were given the set of functional requirements, including the running example. They had the liberty to test their own functional requirements, choose the branches of specification or domain for which risk analysis should be performed, and when to stop the risk analysis. The students followed the process described in Section III to generate the RAG.

The students provided their feedback regarding the ease of use, accuracy of the automation, usability, including a list of risks classified into critical, important,
unimportant, uncommon risks, etc. The questionnaires' results were compared to the results obtained from the Risk Analysis Graph, as shown in Fig. 4.

V. DISCUSSION

A. Overall Observation: We have observed that the students who used RAG were not only able to find common risks such as missing route, walking in the wrong direction, etc., unimportant risks such as warnings which ask them to increase the volume, increase screen brightness, etc., but also were able to identify some critical risks such as falling down, bumping into people, colliding against walls, unexpected object running into the user, etc. and uncommon risks such as oil on the floor, banana peel on the way, etc. while some students who developed and used the questionnaires to find risks have ignored a few critical risks such as low battery indication, faulty voice input due to background noise, walking in a zig-zag fashion in a straight corridor, etc. which we were able to find out by generating multiple RAG’s using different sets of requirements.

B. Threats to Validity: Our evaluation is based on human knowledge, and the decision to generate the RAG using the semi-automated tool may not be accurate all the time. The results of the experimentation included are not real software projects (questionnaires). As our evaluation can be subjective and incomplete, it should be expanded with various subjects (developers, requirements engineers, etc.). Furthermore, the range of experiments and the data obtained was also limited. To try a more diverse range of domains, we do not have sufficient guiding ontology for customizing the model.

VI. CONCLUSION

This paper has presented an ontology-based framework for performing risk analysis by using a Risk Analysis Graph (RAG). The Augmented Reference Model obtained by transforming the Reference Model is illustrated by using a collaborative system as a reference application to validate the strengths and weaknesses of the Risk Analysis framework. More specifically, this paper has presented 1) an ontology, which incorporates crucial concepts such as Agents, Risks, Requirements, Specifications, etc.; 2) the Augmented Reference Model, obtained by transforming the Reference Model to perform risk analysis by negating the events in the environment; 3) A Risk Analysis Graph (RAG) to identify and analyze risks by the negation of logical implication and a couple of cases to identify risks which cannot be obtained by negation. The experimentation, we feel, shows that our approach facilitates the detection of several kinds of risks (common, uncommon, critical, etc.). Apart from these, we feel that we could find critical and unexpected risks using RAG.

There are several lines of future work that we would like to work on. We plan on adding risk prevention and risk mitigation strategies to the risks identified using the RAG. We also plan to develop a constructing algorithm for developing the RAG. There are other domains, such as the Auto-drive domain for Autonomous vehicles, etc., that we would like to extend our work to. We would also investigate more ontologies pertaining to other domains as well. The tool is in its first phase of implementation, and work is being done on adding more features for performing risk identification, risk prevention, and mitigation techniques. For engineers to develop and design their own graphically oriented Risk Analysis Graph (RAG) for identifying risks is also underway. We also plan to include safety and timeliness softgoal and extend our work using a goal-oriented approach.

REFERENCES

Abstract—It is pivotal to have well-specified requirements to eliminate errors at an early stage of the system development life cycle. Some quality standards recommend the use of formal methods – mandate requirements to be expressed in formal notations – to detect errors. However, formal notations are not suitable for non-experts and may not be understood by all the stakeholder. To fix this, bidirectional transformations among requirement representation levels are required to maintain traceability and facilitate the communication of requirements among all the involved parties. This paper reflects on the different formality levels of requirements specifications including: informal, semi-formal, and formal notations. In addition, an automated multi-layer transformation approach is proposed to enable bi-directional transformation among requirements levels.

Index Terms—Requirements engineering, Informal notation, Semi-formal notation, Formal notations, Transformation

I. INTRODUCTION

The complexity of modern systems is rapidly increasing as a result of the incorporation of cutting edge technology in various fields (e.g., automotive, robotics, and Internet-of-Things). These systems have special characteristics over classical systems (e.g., integrating multiple subsystems, scalability, reusability, and stringent requirements measures for: reliability, safety and security, etc.) [1], [2]. In addition, depending on the scope of application, development errors in such systems (e.g., inconsistency, incompleteness, and incorrectness) can lead to catastrophic consequences, severe losses, and hazardous operational failures. Earlier the detection and resolution of such errors, the better quality and control through the development life cycle [3].

Requirements Engineering (RE) is the first phase within the development life cycle [4] and thus contributes greatly to the overall quality of the developed system and achieving a successful and efficient development. The core artefact for this process is the requirements specifications document. There are three levels for representing requirements specifications: informal, semi-formal and formal notations. Each representation level has its own strengths and weaknesses. Informal (natural language) offers the easiest form of communicating requirements but is inherently ambiguous, incomplete, and imprecise. Formal notations are precise and unambiguous but require mathematical background and expertise. And semi-formal notations offer an intermediate trade-off between the two, but is not suitable for all scenarios or domains. Thus, the transformation between these levels of formality is required to maintain fast and efficient communication at different stages of the development between all the contributors in the RE team (e.g., non technical users, formal methods experts, etc) [5]–[7].

In the last three decades, several approaches were proposed for automating the transformation processes. The majority of such approaches focus on transforming requirements from the informal and semi-formal levels to the formal level. This is done to allow formal methods to detect quality issues within the requirements [8], [9]. Yet the problem of transforming requirements from a more formal level of representation to a less formal one is still an open problem. In addition, to the best of our knowledge, very few work has provided a comparison between the different formality levels or discussed their features and the possible transformations between them (e.g., [5], [10]) . To address these gaps, in this paper we provide:

- An analysis of the formality levels for requirements representation and the current state of transformations among them for a better understanding of the problem.
- A multi-layer automated transformation approach from formal notation to semi-formal, then to informal notation.

II. REQUIREMENTS REPRESENTATION FORMALITY LEVELS

Formality levels notations are defined based on the concepts of syntax and semantics visualised in Fig[1] In [11], these concepts are defined from the linguistic perspective and the field of formal language as indicated below:

- Syntax: The syntax of a language is the set of rules that define structured sentences or fragments of the language (e.g., grammatical phrases or sentence structure in natural languages).
Fig. 1. Formality Levels from syntax and semantic perspectives

- Completely defined syntax: The syntax of a language L is completely defined if and only if, for any fragment S, it can be verified whether or not S belongs to L [13].
- Semantic: The semantic of a language is the possible interpretations or meanings of a fragment.
- Completely defined semantics: The semantic of a language is completely defined if and only if, for any given fragment S that belongs to L, there is only one interpretation. This can be confirmed by transforming the language notation into a mathematical system where proofs can be performed [11].

A. Informal Notations

Quality standards [14] define informal notations as a technique with incomplete defined syntax, that is used to specify requirements. Having incomplete syntax also means that the semantics aspect is also incomplete.

From the industry perspective, informal notations have a similar definition where it is defined as free-textual or freestyled requirements [5], [15], [16]. However, there are two fundamental constraints on the use of informal notations: (1) auto-completeness – each individual requirement should be fully understandable, and (2) implied references should be avoided as recommended in [16]. The absence of rules controlling the writing style, makes requirements specification easier but affects the uniformity as it depends on the system engineer’s writing, background and expressiveness skills. Lacking uniformity makes the requirements vulnerable to ambiguities and inconsistencies [17].

The most commonly used informal notation today is Natural Language (NL) (i.e., usually English) [14]. The principle advantage of NL is that it is well known. Thus, it is a good communication method among non-technical users (e.g., customers, stakeholders, etc.), and technical users (e.g., developers, experts, etc.). In addition, there is no need for training to use NL in specifying requirements. This minimises the time needed to create the first project artifact. More details about what are the sections of the specification document, how each section is structured and what are the main elements that should be contained are available here [15].

Despite the advantages, writing requirements in NL has many drawbacks. First, the expressiveness power of informal notation allows requirements documents to contain all kinds of freedom (e.g., ambiguity, contradiction, inconsistency [18]). As NL is inherently ambiguous, different persons may have different interpretations of the same requirement. This may lead to implementing unintended functions or implementing more functions than the intended ones. Having unnecessary functions increases the complexity of the system and consequently the number of errors [19]. In addition, missing a system function would drive the system into incorrect states. Secondly, NL is hard to maintain as the proper grouping of related requirements can not be ensured. Thirdly, due to the free-style of writing, it is hard to verify requirements and make sure that each process has correct input and output.

Current research witnesses a lot of progress towards improving the quality of informal/textual requirements. The existing approaches can be classified into two categories: (1) detecting quality issues (e.g., ambiguity, inconsistency, incompleteness, etc.) in informal requirements – helping engineers refine written requirements– (e.g., [20], [21]), (2) providing defined formats (templates, patterns, boiler-plates, constrained NL) for engineers to utilise while (re-)writing requirements (e.g., [22]).

B. Semi-formal Notations

Semi-formal notations provide an intermediate layer between informal and formal notations. It is a technique for describing requirements with a completely defined syntax and may have incomplete semantics, as defined in most quality standards [14]. From the industry perspective, semi-formal notations are usually either a form of a graphical representation of the system [5], [10], [15] or Constrained Natural Language "CNL" [10], [23] that is developed to minimise ambiguities and improve the readability of the requirements. In fact, there is no contradiction between these definitions because both graphical representations and CNL have completely defined syntax and poorly defined semantics [5], [15].

Graphical representations utilise graphical structures to represent the system [10]. Although not all the requirements issues are eliminated, the effectiveness of graphical representations can be very high. Such representations are popular as they can improve the quality of the requirements with a slight learning curve and give an insight about the system from different points of view (e.g., UML [24], Sysml [25], URN [26], ER-diagrams [27], etc). A graphical representation may be accompanied with some logical languages to raise expressiveness (e.g., OCL is attached to UML in [10]).

CNL is a textual writing with constraints on styling, syntax, vocabulary or mix of them [28]. This limits the engineers to only using the predefined writing style to improve the quality of the written requirements. Commonly, this manifests as boilerplates [29] or Structured English [30] following the defined rule(s). This constrains how the requirements are textually represented by layout and vocabulary (CNL might be impractical if too restrictive [31]).

Semi-formal notations have several advantages: (1) Eliminate, in most cases, issues with the requirements represented in informal notations as the complete syntax limits the expressed variations of a requirement [5]. (2) More comprehensible
than informal notations (detailed descriptions in NL may be confusing when describing complex systems). (3) Graphical representations can give an abstract view of the system providing a better understanding of the system [10]. Visualising the entire system helps find otherwise unnoticed gaps. However, the major drawback is due to the lack of complete semantics—everyone may have their own interpretation [5], [14].

C. Formal Notations

Formal notations are mainly used by formal methods for requirements model verification. They have a completely defined syntax and semantics as defined by the quality standards [14]. Similarly, formal notations are used in the industry as precise notations based on the concept of mathematics to ensure the verification applicability on requirements specifications [15], [32], [33]. Formal notations require strong expertise in mathematics set theory and predicate logic to have the ability to state or understand the formalised requirements of a system [15]. Thus, many (except those with the right expertise) can not easily or clearly understand most of these notations. The most widely used formal notations include (Temporal Logic [34], Z [35], SAL [36], etc.).

A major advantage of formal notations is reducing the development cost and time by discovering errors in the early stages of the software development life cycle. Due to the absence of ambiguity in formal notations, it is reliable to verify whether the system conforms to the specifications or not. Another advantage is improving the quality of the system as the requirements are stated in a precise and consistent manner [32]. Moreover, formal notations may help generate full/partial code [5]. In contrast, non-technical users require extensive time consuming in training to understand formal notations. Large numbers of non-technical users find formal notations very complex [18].

III. FORMALITY LEVELS TRANSFORMATIONS

From the previous section, it is clear that the level of formality of the requirements affects the quality of the represented requirements and the understand-ability among the people involved in the development life cycle (technical and non technical). Thus, maintaining the requirements consistency and traceability is pivotal [5] and can be controlled through transformations. Inspired by the need—in both research and industry—of having viable transformations among the levels of formality [5–7], we outline the existing formality levels transformations as follows:-

Informal notations to Semi-formal notations: The popular direction is to transform informal notations into graphical representations. This can be accomplished by applying a set of linguistic parsing rules on free-textual requirements. Linguistic analysis is a viable technique coping with this type of transformation as illustrated in [23], [44]. These approaches are restricted to and reliant on the accepted subset of NL and the corresponding set of hand-crafted rules, that are domain-specific and can only work for limited scenarios [9]. Alternatively, the combined work proposed in [41] and [40] together can be considered as a more flexible and domain-independent approach that transforms clause-based free-textual requirements into formal language.

Semi-formal notations to Formal notations: This type focuses on transforming either graphical representations or textual-based semi-formal notations to formal notations. It is considered a critical transformation due to the value of formal verification on the derived formal models.

- Textual-based to formal notations:-- most of the existing approaches adopt NLP techniques to parse the defined meta-model of the used textual format to generate the corresponding formal notation (e.g., [45]–[48]). Much progress has been witnessed within this type of transformation. Many different textual-based notations (along with specific parsing techniques) have been proposed to serve a specific domain or a specific type of requirements.

- Graphical representations to formal notations: similarly to the textual-based type of transformation, existing techniques in this category provide a homomorphic mapping between the meta-models of the graphical representations and the target languages (e.g., [32], [49], [50]. In [41], requirement capturing model (RCM) is proposed to enable automatic transformation into various temporal logic based notations, where it is mapped to metric temporal logic and computational tree logic.

Semi-formal notations to Semi-formal notations: In this class, conceptual mapping between the intended notations is formulated to enable transformation [51], [52]. In [53], system requirements capturing model (SRCM) is constructed by aggregating RCMs presented in [41](i.e., each representing one single requirement), to enable compact and integrated system views and quality issues detection in the semi-formal level.

Formal notations to Semi-formal notations: This type of transformation produces a semi-formal notation given the formal one. The most targeted semi-formal notations are graphical based [54]. In this transformation, homomorphic mapping and construction algorithms support the generation of semi-formal diagrams from formal models to visualise complex models [54].

IV. APPROACH

In our previous work [41], we analysed the existing requirements representation models and proposed RCM as a comprehensive semi-formal model supporting behavioural requirements. RCM defines the key requirements elements required for the automatic generation of the corresponding formal notations. We also formulated the mapping rules between the
RCM’s key elements and the metric temporal logic “MTL” formal notation. In addition, we developed a fully automated approach for generating MTL formulas for requirements expressed in RCM. In [40], we extended the work by proposing an automatic NLP-based approach for extracting RCMs from NL-requirements. In this paper we provide requirements transformation in the reverse direction (i.e., MTL to RCM, then RCM to NL-requirements) to maintain consistency among the different levels of formality. Our approach takes a text file containing MTL formulas and a definition file (for interpreting formal symbols) as input, and provides XML and text files for the corresponding generated RCMs and NL-requirements respectively. Figure 2 shows the two transformation layers.

A. MTL to RCM Transformation

We utilise the same mapping rules proposed in [41]. We represent such rules as regular expressions to facilitate matching the underlying formal grammar of the rules. Table I shows the crafted regular expressions mapped to the RCM-MLT mapping formulas. However, regex rules 1, 2, 3, and 7 are crafted according to the formalisation approach proposed in [41]. We apply a bottom-up abstraction approach on the input formula to construct an intermediate table (IT) containing the required details for RCM. The approach consists of five steps shown in Algorithm IV-A each designed to match a specific set of rules against the input formulas to extract specific information.

Table I: Mapping Regex Rules

<table>
<thead>
<tr>
<th>RCM-Role</th>
<th>MTL-Id</th>
<th>Regex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicate Structure 1</td>
<td>1</td>
<td>(w+)(&gt;</td>
</tr>
<tr>
<td>Predicate Structure 2</td>
<td>2</td>
<td>(w+)((w+)+w+)</td>
</tr>
<tr>
<td>Predicate Structure 3</td>
<td>3</td>
<td>(w+)(&gt;</td>
</tr>
<tr>
<td>Pre-Time Element</td>
<td>10:14</td>
<td>$t\leftarrow</td>
</tr>
<tr>
<td>Valid-Time Element</td>
<td>15:19</td>
<td>G(\leftarrow</td>
</tr>
<tr>
<td>In-Time Element</td>
<td>20:24</td>
<td>G(\leftarrow</td>
</tr>
<tr>
<td>Coordinated Elements</td>
<td>7</td>
<td>(\leftarrow</td>
</tr>
<tr>
<td>Scope StartUP</td>
<td>3</td>
<td>G(A-Z) \rightarrow F(A-Z)</td>
</tr>
<tr>
<td>Scope EndUP</td>
<td>6:7</td>
<td>F(A-Z) \rightarrow (P(A-Z) \rightarrow A-Z)</td>
</tr>
<tr>
<td>Scope both</td>
<td>8:9</td>
<td>G(A-Z) \rightarrow (P(A-Z) \rightarrow A-Z)</td>
</tr>
<tr>
<td>Single PreCond</td>
<td>2:3</td>
<td>G(A-Z) \rightarrow A-Z</td>
</tr>
<tr>
<td>Composite PreCond</td>
<td>4</td>
<td>G(A-Z) \rightarrow A-Z</td>
</tr>
<tr>
<td>Action</td>
<td>1:13</td>
<td>G(A-Z) \rightarrow A-Z</td>
</tr>
</tbody>
</table>

In Step 1, each predicate element is identified and represented with a Unique alphabet, where the letter and the corresponding matched group are stored in the IT. Step 2 matches regular expression rules R{4:6} against the MTL formula in an iterative manner to identify time elements. Each identified time element and its type are attached to the corresponding alphabet representing the predicate element in the IT (to keep their composition relation). In addition, the matched elements are replaced with their related predicate alphabets. Step 3 groups the coordinated elements (same type). Step 4 identifies the scope of the main elements. In Step 5, the main elements are identified. The IT is then parsed into RCM. Figure 3 shows a step by step annotation of the transformation algorithm on an example MTL formula along with the constructed IT. The generated RCM is shown in the first column in Figure 4.

B. RCM to NL-Requirements Transformation

This layer consists of two tasks: realisation and sentence structuring. The generated sentences are governed by the following grammar:

Algorithm IV-A: MTL-to-RCM Transformation

States:
R: MTL-to-RCM indexed Regex Rules
Formula: input string representing one formula
Tbl: Table contains extracted information
procedure
Step 1: Identify predicate elements
Formula ← abstractMatchedRegx(R{1:3},Formula)
updateTable(Tbl)
Step 2: Identify attached time elements and their types
Formula ← abstractMatchedRegx(R{4:6},Formula)
updateTable(Tbl)
Step 3: Identify predicate elements with same type
Formula ← abstractMatchedRegx(R{7},Formula)
updateTable(Tbl)
Step 4: Identify Scope elements attached to main elements
Formula ← abstractMatchedRegx(R{8:10},Formula)
updateTable(Tbl)
Step 5: Identify main elements types
Formula ← abstractMatchedRegx(R{11:13},Formula)
updateTable(Tbl)
end procedure

In Step 1, each predicate element is identified and represented with a Unique alphabet, where the letter and the corresponding matched group are stored in the IT. Step 2 matches regular expression rules R{4:6} against the MTL formula in an iterative manner to identify time elements. Each identified time element and its type are attached to the corresponding alphabet representing the predicate element in the IT (to keep their composition relation). In addition, the matched elements are replaced with their related predicate alphabets. Step 3 groups the coordinated elements (same type). Step 4 identifies the scope of the main elements. In Step 5, the main elements are identified. The IT is then parsed into RCM. Figure 3 shows a step by step annotation of the transformation algorithm on an example MTL formula along with the constructed IT. The generated RCM is shown in the first column in Figure 4.
Fig. 5. Structured sentence of the realised RCM in compliance with the elements priority listed in Table II

V. CONCLUSION

In this paper, we provided an insight about the different levels of formality for representing requirements while highlighting the (dis)advantages of each level. We also presented the current state of research for the transformation among these levels. Finally, we proposed a multi-layer transformation approach to bridge the gap of maintaining traceability between formality level and enabling (non-)technical people to understand system requirements.

REFERENCES


Abstract—El Nino-Southern Oscillation (ENSO) is an irregular periodic oscillation in easterly winds and sea surface temperature (SST) over the tropical Pacific Ocean. El Nino and La Nina are warm and cold phases of ENSO. Oceanic Nino Index (ONI) determines ENSO events by calculating the three-month running mean of SST anomalies over the Nino 3.4 region (5°N-5°S and 120°W-170°W). El Nino refers to ONI greater than +0.5°C and La Nina refers to ONI less than -0.5°C for five consecutive months across the east-central equatorial Pacific. ENSO is one of the main drivers of Earth’s inter-annual climate variability, which causes climate anomalies in the form of tropical cyclones, severe storms, heavy rainfall, and droughts. ENSO not only impacts global climate and oceanic conditions but also impacts food production, human health, and economy. Therefore, forecasting ENSO is of great importance. The main contribution of this study is proposing a convolutional long-short term memory that can capture spatial and temporal relationships between ENSO and environmental variables, such as SST, sea level pressure, meridional wind, and zonal wind. This study not only reports forecast accuracy but also quantifies the uncertainty associated with the forecast. Experimental results show that the proposed model improves the forecast accuracy by 14.8%, 10.4%, 11.8%, and 22.2% for lead times of 3, 6, 9, and 12 months, respectively.

Keywords—Spatial-temporal forecast, Bayesian method, Probabilistic forecast, ENSO events, Climate anomalies, Variational inference

I. INTRODUCTION

El Nino-Southern Oscillation (ENSO) is characterized by irregular periodic variation in easterly winds and sea surface temperature over the Central and Eastern Pacific Ocean. It is one of the main drivers of Earth’s interannual climate variability, which often causes a wide range of climate anomalies in the form of heavy rainfalls, severe droughts, heatwaves, unusual tropical storms, and other extreme weather conditions affecting both tropics and subtropics [1]. ENSO transitions between three phases: El Nino, neutral, and La Nina. Typical El Nino conditions in the east-central Pacific Ocean include a) above-average sea surface temperatures, b) weak easterly winds, and c) deep oceanic thermocline than average. Typical La Nina conditions in the east-central Pacific Ocean include a) below-average sea surface temperature, b) strong easterly winds, and c) shallow oceanic thermocline than average. Oceanic Nino Index determines ENSO events by calculating the three-month running mean of sea surface temperature anomalies over the Nino 3.4 region (5°N-5°S and 120°W-170°W). El Nino is observed in the Pacific Ocean if the ONI is greater than +0.5°C for five consecutive months. La Nina is observed in the Pacific Ocean if the ONI is less than -0.5°C for five consecutive months. Together, El Nino and La Nina can not only impact global weather, climate, and oceanic conditions but also food production, human health, economy, and water supply [1]. However, El Nino and La Nina differ from each other in terms of evolutionary patterns and impact on global climate. For instance, a prolonged El Nino or La Nina for more than two years has caused droughts in several regions of the United States, while the transition from El Nino to La Nina or La Nina to El Nino has caused flash floods in North-Eastern regions of Asia. Due to the impact of ENSO on the global climate, it is important to predict these events in advance.

Forecasting ENSO events in the literature can be classified into two types: a) dynamical and b) statistical models. Dynamical models use mathematical equations to describe physical laws governing interactions of atmosphere and ocean for forecasting ENSO. On other hand, statistical models learn patterns from historical data for forecasting ENSO. However, the latter models are often challenged by the complex and nonlinear nature of the ENSO. Machine learning models are statistical models that can extract salient features from high-dimensional data. Linear models such as autoregressive moving average (ARMA) and autoregressive integrated moving average (ARIMA) are too simple to capture the nonlinear and time-varying nature of the ENSO. To overcome the shortcomings of the linear models, artificial neural networks (ANN) were proposed in recent studies [2, 3]. Although ANNs can handle the nonlinearity in the data, they are not designed to explicitly handle the time-sequential dynamic interactions between variables. On the other hand, recurrent neural networks have a mechanism for capturing both nonlinear and time-varying dynamics of multivariate systems. However, RNNs suffer from vanishing gradient problems while backpropagating the error. A variant of RNN, known as long-short term memory (LSTM) was developed to overcome the vanishing gradient problem by...
introducing several gates that would help the model decide what information to keep and what to forget.

Often, climate data exhibits both spatial and temporal autocorrelation, which means data from nearby locations are more similar compared to data from remote locations. Therefore, another challenge of statistical models is how to map the historical data in a meaningful way such that the model can learn spatial and temporal relationships between observed data and ENSO events. A fully connected LSTM can only handle long-term temporal dependencies between input variables, but they are not useful to capture spatial dependencies. Therefore, an extension of LSTM, which is convolutional long-term memory (CLSTM) is introduced in [4] for precipitation forecasting. CLSTM replaces full connections in input and hidden layers of LSTM by convolution windows. In a recent study, ENSO is formulated as a spatial-temporal sequence forecasting problem, in which both input and output are sea surface temperature sequences [5]. However, their study only reported the forecast accuracy but did not quantify the uncertainty associated with the forecast. Estimating forecast uncertainty not only makes the forecasts reliable but also helps the decision-makers in the field to take appropriate actions.

In this study, spatial and temporal features of ENSO are derived by embedding ENSO predictors into a grid space, which fully expresses the spatial and temporal relationships with ENSO. Then these spatial-temporal features are fed to CLSTM to forecast ENSO events at three-month intervals up to one year. This study not only reports the forecast accuracy but also quantifies the uncertainty of the forecast.

II. LITERATURE REVIEW

Different machine learning and statistical analyses have been applied in literature for predicting environmental and urban phenomena [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. Forecasting ENSO in the literature is explored in this section under two categories: (a) deterministic methods and (b) probabilistic methods.

A. Deterministic methods

There have been many reviews done on ENSO predictability in recent years. Wu et al. [16] also used ANN and support vector regressor (SVR) to forecast sea surface temperature anomalies over the entire Tropical Pacific region with sea level pressure and sea surface temperature as predictors at a lead time 3-15 months. Ham et al. [17] used transfer learning to train a CNN first on historical data and subsequently on reanalysis data from 1871 to 1973. The forecasts were made for the years 1984 to 2017 and the correlation skill of CNN during these years is superior to that of state-of-art dynamical forecast systems. In a recent review, Dijkstra et al. [18] detailed the application of machine learning algorithms and their role in improving the prediction skill of ENSO.

B. Probabilistic methods

Recent studies use hybrid models by combining autoregressive integrated moving average and an ANN to predict the [19]. For a lead time up to six months, their model performed slightly higher than CSFv2 [20]. Whereas, for lead times beyond six months the prediction results are similar to that of shorter lead times. Another example of the hybrid model for forecasting ENSO events can be found in [21]. A suite of statistical and dynamical models [20, 22, 23, 24] are combined using the Bayesian model averaging. The weights for the models are derived using the expectation-maximization algorithm. The dynamical models, for example, CFSv2 [20], provides a single model ensemble forecast. It can give an estimation of the predictive uncertainty by making an ensemble of forecasts. Recent studies estimate the predictive uncertainty of forecasts, using an ensemble of standard neural networks (SNNs), called a deep ensemble (DE) [25], which can provide the mean and standard deviation of a Gaussian distribution instead of point estimates. Another probabilistic approach based on the Bayesian neural network is proposed in [26], which can give an estimate of the predictive uncertainty. However, the evaluation window for their model is very short (2015-2016). Peter et al. [27] experimented with the gaussian density neural network and quantile regression neural network to quantify the uncertainty of the forecasted ENSO. For the test period 1982-2001, the forecast accuracy is above 0.7 for 12 months lead time but drops to 0.4 for six months lead time for 2002-2017.

In this study, we propose CLSTM that can capture spatial and temporal relationships between ENSO and ENSO predictors that contribute in forecasting ENSO. The main difference between our work and previous works is that we not only report forecast accuracy but also quantifies uncertainty associated with the forecast.

III. DATA DESCRIPTION

ENSO events can be determined by forecasting ONI. The predictors used in this study are sea surface temperature (t: °C) sea level pressure (p: Nm⁻²), zonal wind speed (u: m s⁻¹), and meridional wind speed (v: m s⁻¹). The information about these variables is gathered from National Oceanic and Atmospheric Administration [28] and National Center for Environmental Prediction Reanalysis 1 [29]. The data are recorded at 500 different locations extending from (5°N-5°S and 120°W-170°W) from 1950-2019. The temporal resolution of the dataset is 1 month, and the spatial resolution of the dataset is 1° latitudinal and 1° longitudinal distance. Therefore, the dataset consists of a total of 420,000 records. Each record in the dataset represents the average of the three consecutive months. For example, the average sea surface temperature for three consecutive months (January, February, and March) is assigned.
to January. All features (predictors) are normalized using standard scalar within the range of [0,1].

Figure 1 illustrates the variations of ONI from 1950-2019. The red and green lines are reference lines for El Nino and La Nina. Any value beyond the reference lines corresponds to El Nino or La Nina. From the figure, it is inferred that El Nino occurred more frequently than La Nina in the study period. The strongest El Nino recorded was in the year 2015, with a magnitude close to 3, and the strongest La Nina recorded was in the year 1973, with a magnitude close to -2. The data from 1950-2013 is used for training, while the data from 2013-2019 is used for testing.

IV. METHODOLOGY

The presence or absence of ENSO events spans across the Pacific region, but not at a single point in the Pacific region. Relating this with the dataset used in this study, all the data points belonging to the same time have the same output irrespective of their geographical location. This also means that the data points from the same time should be fed as a single input to the prediction model rather than feeding them as different inputs into the prediction model. In this way, the number of samples in the dataset is 840 (70 years multiplied by 12 months). We position ENSO predictors that belong to the same time on spatial grids, where each grid is for each predictor. These spatial grids are arranged next to each other to form a spatial feature map, and this spatial feature map is fed to the prediction model to determine ONI.

LSTMs are suitable for sequential prediction problems as they can remember the state of the previous time step with the help of memory gates. LSTM consists of three gates: input gate, forget gate, and output gate. With these gates, LSTM decides what information to store inside the memory unit and what information to throw away. More information about LSTM can be found in [30]. The flow of information through gates of LSTM is given by the following equations.

\[
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
H_t = o_t \odot \tanh(c_t)
\] (1)

Here \(i_t, f_t, o_t\) represents the input, forget, and output gates respectively, \(W\) with any suffix represents weight matrices, \(b\) with any suffix represents bias vectors and \(\odot\) represents Hadamard product. Though LSTM has shown promising performance in sequence prediction problems, the full connections in input and hidden layers of LSTM cannot encode spatial information and therefore underperforms for spatial-temporal prediction problems. CNN has outperformed other neural network architectures in image processing and computer vision due to its ability to capture pixel dependencies in the image. Unlike LSTM, CNN does not have memory gates to remember the state of previous time steps and therefore underperforms for sequential data.

A. CLSTM

In a recent study, Shi et al. [4] introduced CLSTM for a spatial-temporal prediction problem. CLSTM combines the advantages of CNN and LSTM by replacing the full connections in the input and hidden layers of LSTM with convolutional windows. With this architecture, CLSTM can capture both spatial and temporal dependencies between predictors and the target variable. Therefore, we employ CLSTM in this study to forecast ONI. The flow of information through the gates of CLSTM is as follows.

\[
i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t = f_t \odot c_{t-1} + i_t \odot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
H_t = o_t \odot \tanh(c_t)
\] (6-9)

Here * represents convolution operation and \(\odot\) represents Hadamard product. The original CLSTM proposed in [4] follows encoding-forecasting structure for spatial-temporal sequence-to-sequence prediction. The dimensionality of the target is the same as the input. However, the encoding-forecasting structure cannot be implemented in this study due to different input and target dimensions. The architecture of the CLSTM consists of a 2D-CLSTM layer, a dropout layer, and three fully connected dense layers. The CLSTM layer consists of 32 filters, each filter with a size of 5x5. The size of the filter and the number of filters is determined using hyperparameter tuning. A dropout value of 0.5 means, only 50% of the output from the CLSTM layer is used as an input to successive layers of the network. Dropout avoids overfitting of training data. The first two layers of the fully connected layers consist of 128 and 64 neurons respectively, and the third layer (output layer) has as many neurons as the lead time. For example, the architecture for a lead time of 3 months has three neurons in its output layer.

B. Optimizing Model Parameters using Variational Inference

CLSTM provides the ENSO forecast when the model parameters \(\omega\) are known. To quantify the uncertainty associated with the forecasts, we need to obtain the probability density function of ONI. This can be achieved when the true posterior distribution of model parameters given the observed data is known. Thus, the posterior distribution of model parameters according to the Bayesian rule is given as follows.

\[
P(\omega | X, Y) = \frac{P(X,Y|\omega)P(\omega)}{\int P(X,Y|\omega)P(\omega) d\omega}
\] (11)

Here \(P(\omega)\) is the prior model parameters, \(P(\omega | X, Y)\) is the true posterior distribution of model parameters given observed data, \(P(X,Y|\omega)\) is the likelihood of observed data. However true posterior is generally intractable due to the multidimensional integrals in the denominator of (11). Variational inference (VI) is a technique used to approximate the posterior of model parameters \(P(\omega | X, Y)\) by minimizing the distance between variational distribution \(q_\phi(\omega)\) and true posterior. This is achieved by minimizing the Kullback-Leibler divergence (KL-divergence) between the variational distribution and the true posterior. Minimizing KL-divergence is the same as
maximizing the evidence lower bound (ELBO) concerning the variational parameters. Therefore, the final objective is to obtain the optimal distribution of model parameters by maximizing ELBO. The same can be formulated as follows.

\[ ELBO = \mathbb{L}(q) - KL(q_\omega(\omega)||P(\omega)) \]  

(12)

Here the first term \( \mathbb{L}(q) \) is called expected log-likelihood and the second term is the KL distance between a variational distribution and true distribution of model parameters. Maximizing the first term would result in variational distribution \( q_\omega(\omega) \) that explains the distribution of data well. Maximizing the second term would help the model from overfitting. The variational distribution \( q_\omega(\omega) \) is assumed to be Gaussian with a predefined probability \( \rho \) and the standard deviation \( \sigma \). The first term in (12) can be approximated by a Monte Carlo estimate using the mini-batches chosen randomly from the full dataset \([X, Y]\) as follows.

\[ ELBO \sim -\frac{N}{M} \sum_{m=1}^{M} ||Y^m - f_{\omega}^q(X^m)||^2 - KL(q_\omega(\omega)||P(\omega)) \]  

(13)

It is also assumed that prior follows a normal distribution with zero mean and unit variance \( \mathcal{N}(0, I) \). Therefore, the KL-divergence can be approximated as L2 regularization over the variational parameters \( \theta \). Once the optimal value of the variational distribution is obtained, the same can be used to approximate the true posterior. Now, given a test sample \( x^* \), the probability of forecasting its output \( y^* \) is given by (14).

\[ P(y^*|x^*, X, Y) = \int P(y^*|f_{\omega}^q(x^*)) q_\omega(\omega) d\omega \]  

(14)

To verify the performance of CLSTM, machine learning models, such as SNN, LSTM, CNN, and gaussian process regressor (GPR) are employed as baselines. The network settings used for baselines are as follows. The structure of SNN consists of three fully connected layers. The first two layers of SNN have 100 nodes each, followed by a third layer with as many neurons as the lead time. Similar to SNN, the structure of LSTM consists of three fully connected layers. The first two fully connected layers consist of 128 and 64 neurons respectively. The third fully connected layer has as many nodes as the lead time. Unlike CLSTM, where a sample is spatial maps of ENSO predictors of the same time, a sample in SNN and LSTM is ENSO predictors of a single geographical coordinate. Therefore, we need to train multiple LSTMs for data points recorded at the same time. The structure of CNN consists of two convolutional layers followed by a max-pooling layer after each convolutional layer and three fully connected dense layers. The first convolutional layer has 64 filters, each filter of size 5x5. The second convolutional layer has 32 filters, each filter of size 3x3. The first two fully connected layers have 100 nodes each, followed by a third layer with as many neurons as the lead time. For all neural network-based baselines, Relu is the activation function used in the hidden layers. GPR is a probabilistic method based on Bayesian theory. Multiple GPR models are trained for data points recorded at the same time. Each model corresponds to each geographical coordinate. In all baselines, mean squared error (MSE) is the loss function employed for error backpropagation.

V. RESULTS AND DISCUSSIONS

The results of hyperparameters are discussed as follows. The size of the filter and the number of filters determine the spatial information captured by the model.

An increase in filter size increases the model’s ability to capture multiple representations of the spatial dependencies. However, with more filters, the complexity of the model increases and they overfit training data quickly. Figure 2 illustrates the performance of CLSTM for various filter sizes and lead times. From the figure, it is evident that the \( R^2 \) value increases when the number of filters is changed from 16 to 32 and decreases when the number of filters is changed from 32 to 64. Since the performance of CLSTM with 32 filters is greater, the optimal choice for the number of filters is 32.

As the filter size increases, CLSTM can capture a wide range of spatial dependencies between variables. Generally, odd-sized filters are used for modeling. The performance of CLSTM for various sizes of the filter is illustrated using Figure 3. A filter of size \( 1 \times 1 \) means no spatial information is encoded for forecasting. For shorter lead times, filter size doesn’t seem to have much effect on the model performance. For lead time beyond 6-months, the filter size of \( 5 \times 5 \) performs best. From the results, we can conclude that for longer lead times, the model performs best when provided with larger amounts of spatial information.

The forecasting skill of all the experiments conducted in this study is verified using three metrics: root mean squared error (RMSE), mean absolute error (MAE), and coefficient of
determination ($R^2$). The metrics are defined by the following equations.

\[
RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}
\]  

(15)

\[
MAE = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i|
\]  

(16)

\[
R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2}
\]  

(17)

Here, $y_i$ and $\hat{y}_i$ denote the actual and predicted values of the $i^{th}$ sample in the testing set, $\bar{y}$ and $N$ denote the mean and total number of samples in the testing set. The smaller is the value of RMSE and MAE, the better is the model performance. $R^2$ reaches its best at one and worst at zero.

Table I shows the comparison of different models for various lead times. The proposed model, CLSTM, outperforms other models in terms of $R^2$, RMSE, and MAE for various lead times. Particularly, $R^2$ values are close to one for lead times up to three months. Although the value of $R^2$ gradually drops for a lead time beyond six months, the proposed model is still superior when compared with others. CNN and LSTM almost performed equally in terms of $R^2$ for lead time up to six months. The better performance of CNN is due to the ability to capture spatial relationships between ENSO predictors and ENSO. The better performance of LSTM is due to the ability to capture temporal relationships between ENSO predictors and ENSO. The performance of CNN and LSTM drops beyond six months. SNN performed moderately for a lead time of three months. Beyond three months, the performance of SNN drops due to the lack of proper handling of spatial-temporal data. GPR achieved lower $R^2$ and greater RMSE, and MAE scores. Experimental results demonstrate the ability of CLSTM in capturing both spatial and temporal relationships between ENSO predictors and ENSO. The results indicate the forecast accuracy is improved by 14.8%, 10.4%, 11.8%, and 22.2% for lead times of 3, 6, 9, and 12 months respectively.

TABLE II. ACCURACY OF MODELS IN FORECASTING PROBABILISTIC DENSITY OF ONI

<table>
<thead>
<tr>
<th>Lead time in months</th>
<th>Model</th>
<th>RMSE</th>
<th>MAE</th>
<th>$R^2$</th>
<th>RMSE</th>
<th>MAE</th>
<th>$R^2$</th>
<th>RMSE</th>
<th>MAE</th>
<th>$R^2$</th>
<th>RMSE</th>
<th>MAE</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GPR</td>
<td>0.739</td>
<td>0.858</td>
<td>1.29</td>
<td>1.87</td>
<td>2.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>CLSTM</td>
<td>0.726</td>
<td>0.844</td>
<td>1.08</td>
<td>1.66</td>
<td>1.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The CRPS scores for GPR and CLSTM are reported in Table 2. From the table, it is clear that CLSTM has the lowest CRPS score value for all lead times. The results conclude that the proposed model, CLSTM-BVI not only provides an accurate forecast but also a reliable probability density of the ONI.

![Figure 4. Forecasted versus observed ONI](image)

To estimate the forecast uncertainty obtained by CLSTM, Figure 4 is plotted for forecasted ONI and observed ONI. The light blue shade represents the 95% prediction interval, the light gray shade represents the 70% prediction interval. The blue solid line represents the forecasted ONI, and the dashed red line represents the observed ONI. As shown in the figure, the forecasted ONI follows the observed ONI, demonstrating the accuracy of CLSTM. Also, the observed ONI falls within the 95% prediction interval which means the proposed model is 95% confident that future values of ONI are contained within the 95% prediction interval.

VI. CONCLUSIONS AND FUTURE DIRECTIONS

Experimental results demonstrate that CLSTM outperforms other models in terms of $R^2$, RMSE, and MAE. The $R^2$ value of CLSTM is close to one for shorter lead times (< 6 months), indicating the superiority of CLSTM in capturing spatial and temporal relationships between ENSO predictors and ENSO. However, the $R^2$ value gradually drops to 0.52 for longer lead times (> 6 months). An overall improvement in the forecast accuracy of CLSTM is observed. Experimental results indicate that the forecast accuracy of the proposed model improves by 14.8%, 10.4%, 11.8%, and 22.2% for lead times of 3, 6, 9, and
12 months respectively. The low CRPS value proves the superiority of the proposed model when compared with GPR. In addition to the forecast, the uncertainty associated with the forecast is also shown. The experimental results demonstrate that the CLSTM is 95% confident that the future values of ONI are contained within the 95% prediction interval. Therefore, the proposed model not only improves the forecast accuracy but also quantifies uncertainty associated with the forecast.

In future work, other environmental variables, such as warm water volume, westerly wind bursts, and upper ocean heat content can be added as predictors for forecasting ENSO. A robust framework can be built by adding more layers to the CLSTM network. Advanced methods can be proposed by integrating the existing methods with numerical weather prediction models.

REFERENCES


Recurrent Neural Graph Collaborative Filtering

Beichuan Zhang, Zhijiao Xiao* and Shenghua Zhong
College of Computer Science and Software Engineering, Shenzhen University, China
zhangbeichuan2019@email.szu.edu.cn, {cindyxzj, csshzhong}@szu.edu.cn

Abstract—Collaborative Filtering (CF) is a prevalent technique in recommender systems. Substantial research focuses on learning the embedding of users and items via exploiting past user-item interactions. Recent years have witnessed the boom of Graph Convolutional Networks (GCNs) on CF. Performing graph convolution iteratively, GCN-based models concatenate/average/sum all outputs from different graph convolution layers to generate the embeddings of users and items. Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights and the weights are related to sequential dependencies from precursor nodes. To resolve the aforementioned problems, in this work, we present a new model, Recurrent Neural Graph Collaborative Filtering (RNGCF), which proposes a sequential dependency construction module to adaptively generate the embeddings. Specifically, the module applies a gated recurrent unit (GRU) to learn the sequential dependencies from precursor nodes and an adaptive gated unit (AGU) to adaptively construct the embeddings based on the sequential dependencies. Extensive experiments on three benchmark datasets show that our model outperforms state-of-the-art models consistently. Our implementation is available in PyTorch 1.

Keywords—Collaborative Filtering, Graph Convolutional Network, Recurrent Neural Network, Recommender System

1. INTRODUCTION

To alleviate information overload on the web, recommender systems have been widely applied to many online services such as E-commerce and advertising [1], [2]. The goal of recommender systems is to predict whether a user will interact with an item, e.g., click, rate, purchase. As an effective solution, CF achieves the prediction via exploiting past user-item interactions. In CF, substantial research focuses on learning the embeddings of users and items to predict a user’s preference for an item based on the similarity of the embeddings [2], [3]. As many user-item interaction data show graph structures, GCNs [4], [5] have been widely applied to CF [6], [7]. In GCNs, one graph convolution layer aggregates the features from nodes that are one hop away in the graph. This implies that the output of the $k$-th graph convolution layer aggregates the features from nodes that are $k$ hops away in the graph [8]. Performing graph convolution iteratively, existing GCN-based works on CF concatenate/average/sum the all outputs from different graph convolution layers to generate the embeddings of users and items for combining all features from different hops [6], [7], [9].

1https://github.com/Book1996/RNGCF
DOI reference number: 10.18293/SEKE2021-015

Fig. 1. An example of the paths rooted at $u_1$. Nodes in different color circles belong to different hops.

Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights and the weights are related to the sequential dependencies from precursor nodes. In a user-item interaction graph, the explanations behind a progressive path unveil the nodes from different hops have different roles in reflecting the preference of users. For example, a bipartite user-item graph is denoted as Figure 1 left, where the edges between user $u$ and item $i$ are observed interactions. Examining the paths rooted at $u_1$ as Figure 1 right, as discuss in papers [6], [10], the first-hop path $u_1 \rightarrow \{i_1, i_2, i_3\}$ shows $u_1$’s preference is related to the features of $i_1, i_2$ and $i_3$; the second-hop path $u_1 \rightarrow \{i_1, i_2, i_3\} \rightarrow \{u_2, u_3\}$ shows the behavioral similarity between $u_1$ and $\{u_2, u_3\}$, because $u_2$ and $u_3$ have interacted with the items that $u_1$ also has interacted with; the third-hop path $u_1 \rightarrow \{i_1, i_2, i_3\} \rightarrow \{u_2, u_3\} \rightarrow \{i_4, i_5\}$ shows $u_1$ may be interesting in $i_4$ and $i_5$, since $u_1$’s similar users $\{u_2, u_3\}$ have consumed $i_4$ and $i_5$. This example shows the neighbor nodes from different hops have different roles in reflecting the preference of $u_1$. Furthermore, the roles of the nodes are related to the precursor nodes. In other words, the features of the nodes that are different hops away would have different weights to generate the embedding of $u_1$. Ignoring this fact, the existing works have several limitations: 1) Without adaptively constructing the embeddings could lead to the suboptimal representations of nodes; 2) Suppose there are two paths, the two paths are composed of the same nodes. But the orders of the nodes in the two paths are different. The average operations in the existing works may confuse the two paths to make the embeddings excessively similar.

To address the aforementioned problems, we propose a new GCN-based model, Recurrent Neural Graph Collaborative Filtering (RNGCF). RNGCF inputs a user’s ID and a candidate item’s ID and apply GCNs to output the user’s preference for the candidate item. Different from GCN-based previous methods, we propose a sequential dependency construction module...
to adaptively construct the embeddings of users and items. The construction module applies a GRU [11] to learn the sequential dependencies from precursor nodes and an adaptive gated unit (AGU) to adaptively construct the embeddings of users and items based on the sequential dependencies. We perform extensive experiments on three standard large real-world CF datasets, and the results clearly show our model can achieve better performance than state-of-the-art methods. To justify the designs in our model, we further conduct ablation studies on RNGCF. The results show each component of RNGCF has contributions to performance. To summarize, this work makes the following main contributions:

- We point out it is important to adaptively construct the embeddings of users and items based on the sequential dependencies from precursor nodes for our task.
- We propose a new GCN-based model RNGCF, which applies a gated recurrent unit (GRU) and an adaptive gated unit (AGU) to adaptively construct the embeddings of users and items based on the sequential dependencies.
- We demonstrate our model can achieve state-of-the-art results by extensive experiments on three standard large real-world CF datasets.

II. RELATED WORK

A. Model-based CF Methods

Collaborative Filtering (CF) is a prevalent technique in modern recommender systems. Among the various CF methods, item-based methods estimate a user’s preference for an item via measuring the item’s similarities with the items in her/his interaction history [12]. User-based methods estimate a user’s preference for an item via finding similar users to the current user and then recommend the items in her/his similar users’ interaction history. Other research focuses on learning the embedding of users and items by reconstructing historical user-item interactions. For example, Matrix Factorization (MF) [13] reconstructs historical user-item interactions via conducting inner product between the embeddings of users and items. BPR-MF [3] presented a pairwise ranking loss to optimize MF. NCF [14] pointed out that the inner product in MF had an inherent limitation and replaced the inner product with a multiple-layer perceptron (MLP). Another type of CF method does not project the IDs of each user as embedding vectors, the methods consider historical items of a given user as the embedding of the user [15], [16]. For example, SVD++ [15] regarded the weighted average of the embeddings of historical items as the embedding of users. With the development of attention mechanism, ACF [16] proposed to adaptively learn the weight of each historical item. We focus on how to learn the embedding parameters of users and items.

B. Graph-based CF Methods

Recent years have witnessed the boom of Graph Convolutional Networks (GCNs) [4], [5], [17]–[19]. As many real-world datasets in recommender systems show graph structures, researchers attempt to adopt GCNs for recommendation [1], [7], [20]. For example, Graph Convolutional Matrix Completion (GCMC) [21] proposed a graph auto-encoder framework to resolve matrix completion tasks. PinSage [1] combined random walk and graph convolution to handle recommendation tasks with billions of items and hundreds of millions of users. HOP-Rec [22] introduced confidence weighting parameters to incorporate graph convolution and random walk. NGCF [6] devised a new graph convolution layer to encode more collaborative signals into the embeddings of users and items. LightGCN [7] proved that the feature transformation and the nonlinear activation in NGCF [6] were useless in the recommendation task that only uses the IDs of users and items. NGAT4rec [9] employed a novel neighbor-aware graph attention layer that assigned different attention coefficients to the different neighbors of a given node. Although the previous methods have been proven effective, the pooling operations in the previous methods fail to consider the outputs from different graph convolution layers have different weights. Our GCN-based method applies a GRU and an AGU to adaptively construct the embeddings of users and items. There are other approaches that combine GCNs and recurrent architectures on different domains. For example, Evolving Graph Convolutional Networks (EGCN) [23] used a GRU to capture the dynamics of the graph sequence. For traffic prediction, Temporal Graph Convolutional Network (T-GCN) [24] adopted GCN to capture spatial dependence and GRU to capture temporal dependence.
III. METHOD

In this section, we introduce the architecture of our model in detail as shown in Figure 2, which includes an embedding layer, stacked graph convolution layers, a sequential dependency construction module.

A. Problem Formulation

In many real-world recommendation scenarios, user implicit data (e.g., click, rate, purchase) are more common than explicit data (e.g., ratings). Following previous GCNs based models [6], [7], we focus on implicit data. Suppose we have N users, M items and an user-item interaction graph G. Based on G, we can define $R \in \{0, 1\}^{N \times M}$ as an implicit feedback interaction matrix. Entry $R_{ui}$ in the interaction matrix R indicates whether user $u$ interacted item $i$, which can be defined as follow:

$$R_{ui} = \begin{cases} 1 & \text{if (u,i) interaction is observed} \\ 0 & \text{otherwise} \end{cases}$$  

Typically, most of the entries in the interaction matrix $R$ are unobserved (0). Based on the interaction matrix $R$, the aim of our task is to predict preference scores for unobserved entries in $R$.

B. Embedding Layer

Following most of recommender models [3], [25], [26], we create an embedding table $E^0 \in \mathbb{R}^{(M+N) \times d}$ to project the IDs of $N$ users and $M$ items into initial vector representations, where $d$ denotes the embedding size:

$$E^0 = [e^0_{u1}, e^0_{u2}, \ldots, e^0_{uN-1}, e^0_{uN}, e^0_{i1}, e^0_{i2}, \ldots, e^0_{iM-1}, e^0_{iM}]$$  \hspace{1cm} (2)

We represent the initial embeddings of user $u$ and item $i$ by $e^0_u \in \mathbb{R}^d$ and $e^0_i \in \mathbb{R}^d$, respectively. It is worth noting that these embeddings serve as the initial states for user $u$ and item $i$, but are not embeddings that are used to predict. Thus, the superscript of the embeddings is 0.

C. Graph Convolutional Layer

Following the prior works [7], [10], we remove the two most common designs in GCNs: the feature transformation and the non-linear activate function. The stacked graph convolution layer in RNGCF is defined as:

$$e^k_u = \frac{1}{|N_u|+1} e^{(k-1)}_u + \sum_{i \in N_u} \frac{1}{|N_u|+1} \sqrt{\frac{1}{|N_u|+1}} e^{(k-1)}_i, \hspace{1cm} (3a)$$
$$e^k_i = \frac{1}{|N_i|+1} e^{(k-1)}_i + \sum_{u \in N_i} \frac{1}{|N_i|+1} \sqrt{\frac{1}{|N_i|+1}} e^{(k-1)}_u, \hspace{1cm} (3b)$$

where $N_u$ denotes the set of items that are interacted by user $u$, $N_i$ denotes the set of users that interact with item $i$. $e^k_u \in \mathbb{R}^d$ and $e^k_i \in \mathbb{R}^d$ respectively represent the new embeddings of user $u$ and item $i$ after $k$ graph convolutional layers.

D. Sequential Dependency Construction Module

1) Gated Recurrent Unit: After $K$ graph convolutional layers, we can get two sequences $S_u = \{e^1_u, \ldots, e^K_u\}$, $S_i = \{e^1_i, \ldots, e^K_i\}$ for user $u$ and item $i$, respectively. At present, the most widely used neural network models for processing sequence are the recurrent neural networks (RNNs) and Transformer [27]. Compared with LSTM [28] and Transformer [27], the GRU [11] has a relatively simple structure. Since LightGCN has proved that excessive parameters and nonlinear structures have no positive effect on the effectiveness of GCN based models on our task. Thus, we choose GRU to exploit the sequential dependencies in $S_u$ and $S_i$:

$$c_u = GRU(e^1_u, \ldots, e^K_u), \hspace{1cm} (4a)$$
$$c_i = GRU(e^1_i, \ldots, e^K_i), \hspace{1cm} (4b)$$

We regard $c_u \in \mathbb{R}^d$ and $c_i \in \mathbb{R}^d$ as the sequential dependencies in $S_u$ and $S_i$, respectively. GRU: $\mathbb{R}^{K \times d} \rightarrow \mathbb{R}^d$.

2) Adaptive Gated Unit: In order to fully consider the sequence dependency and the original graph convolution output, we propose an adaptive gated unit (AGU) to adaptively construct the final embeddings of users and items based on the sequential dependencies, which can be defined as:

$$e^\prime_u = \sigma(W c_u + b) \odot e^K_u + c_u, \hspace{1cm} (5a)$$
$$e^\prime_i = \sigma(W c_i + b) \odot e^K_i + c_i, \hspace{1cm} (5b)$$

where $W \in \mathbb{R}^{d \times d}$, $b \in \mathbb{R}^d$ are learnable parameters. $\odot$ is the element-wise product. $\sigma$ is Sigmoid function.

E. Message and Node Dropout

Following the prior works [6], [21], we adopt two dropout techniques in RNGCF: message dropout and node dropout. Specifically, we apply the node dropout to randomly drop out some observed interactions. We apply the message dropout to drop out elements in Equation 4 and 5, which are updated as:

$$c_u = GRU(Dropout(e^1_u, \ldots, e^K_u)), \hspace{1cm} (6a)$$
$$c_i = GRU(Dropout(e^1_i, \ldots, e^K_i)), \hspace{1cm} (6b)$$
$$e^\prime_u = Dropout(\sigma(W c_u + b) \odot e^K_u + c_u), \hspace{1cm} (6c)$$
$$e^\prime_i = Dropout(\sigma(W c_i + b) \odot e^K_i + c_i). \hspace{1cm} (6d)$$

F. Model Prediction

The model prediction is defined as the inner product of the final embeddings of user $u$ and item $i$.

$$\hat{y}_{ui} = (e^\prime_u)^T e^\prime_i, \hspace{1cm} (7)$$

where $\hat{y}_{ui}$ is used as the ranking score for recommendation.

G. Model Training

As most of previous methods [7], [9], we employ Bayesian Personalized Ranking (BPR) loss [3] to optimize RNGCF. The objective function can be defined as:

$$L_{BPR} = - \sum_{u=1}^{N_u} \sum_{i \in N_u} \sum_{j \not\in N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||E^0||^2, \hspace{1cm} (8)$$

where $N_u$ denotes the set of items that user $u$ interacted with. $\lambda$ controls the L2 regularization strength to prevent overfitting. $\sigma$ is Sigmoid function.
### IV. EXPERIMENTS

#### A. Experimental Settings

1) **Baselines:** NGCF [6] has shown to outperform several methods including GC-MC [21], PinSage [1], NeuMF [14], and HOP-Rec [22]. As the comparison is done on the same datasets under the same evaluation protocols, we do not further compare with these methods. To verify the effectiveness of our approach, we compare it with the following baselines:

- **MF** [3]: Matrix Factorization (MF) directly embedded user/item IDs as vectored representations and modeled user-item interaction with inner product.
- **NGCF** [6]: NGCF adopted three GCN layers on the user-item interaction graph to encode more neighbors’ information into the embeddings of users and items.
- **LightGCN** [7]: LightGCN proved nonlinear transformation contributed little to the performance of NGCF.
- **DGCF** [10]: DGCF disentangled the representations of users and items at the granularity of user intents since a user generally had multiple intents to adopt certain items.
- **NIA-GCN** [2]: NIA-GCN proposed a cross-depth ensemble layer to preserve the relational information in neighborhood.
- **NGAT4rec** [9]: NGAT4rec generated the embeddings of neighbors according to the corresponding attention coefficients.

2) **Dataset Description:** To keep the comparison fair, we conduct experiments on three benchmark datasets: Gowalla, Yelp2018, and Amazon-Book. The three datasets are exactly as same as the LightGCN [7] and DGCF [10] paper used. For each dataset, 80% of historical interactions of each user are selected to constitute the training set, and the remaining historical interactions are treated as the test set. Yelp2018 is about local businesses like restaurants and bars. Amazon-Book is about the businesses like books. Gowalla contains user-venue check-in information from a location-based social network. The statistics of datasets are summarized in Table I.

#### B. Performance Comparison

Table II shows the best performance of all methods on three datasets. And Figure 3 shows the training curves of RNGCF and LightGCN, which are evaluated by training loss and testing recall@20 per 10 epochs on Yelp2018 and Amazon-Book. Our method RNGCF achieves significant improvements over all methods across three datasets. In particular, RNGCF’s relative improvements over the strongest baselines w.r.t. Recall@20 are 5.53%, 4.76%, and 24.72% in Gowalla, Yelp2018, and Amazon-Book, respectively. This demonstrates the high effectiveness of our model. Across the three datasets, we find that the improvements on Amazon-Book are much more than that on the others. Compared with other datasets, Amazon-Book is the sparsest dataset. This suggests that RNGCF may be more suitable for sparse datasets.

#### Table I. Statistics of experimented data.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>User#</th>
<th>Item#</th>
<th>Interaction#</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gowalla</td>
<td>29,858</td>
<td>40,981</td>
<td>1,027,370</td>
<td>0.00084</td>
</tr>
<tr>
<td>Yelp2018</td>
<td>31,668</td>
<td>38,048</td>
<td>1,561,406</td>
<td>0.00130</td>
</tr>
<tr>
<td>Amazon-Book</td>
<td>52,643</td>
<td>91,599</td>
<td>2,984,108</td>
<td>0.00062</td>
</tr>
</tbody>
</table>

#### Table II. Overview performance comparison. Bold scores are the best and underlined scores are the second best.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon-Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
</tr>
<tr>
<td>MF</td>
<td>0.1388</td>
<td>0.1291</td>
<td>0.0533</td>
</tr>
<tr>
<td>NGCF</td>
<td>0.1629</td>
<td>0.1355</td>
<td>0.0579</td>
</tr>
<tr>
<td>NIA-GCN</td>
<td>0.1726</td>
<td>0.1358</td>
<td>0.0599</td>
</tr>
<tr>
<td>LightGCN</td>
<td>0.1860</td>
<td>0.1554</td>
<td>0.0660</td>
</tr>
<tr>
<td>DGCF</td>
<td>0.1865</td>
<td>0.1562</td>
<td>0.0670</td>
</tr>
<tr>
<td>NGAT4rec</td>
<td>0.1855</td>
<td>0.1534</td>
<td>0.0675</td>
</tr>
<tr>
<td>RNGCF</td>
<td><strong>0.1944</strong></td>
<td><strong>0.1628</strong></td>
<td><strong>0.0707</strong></td>
</tr>
</tbody>
</table>

%Impro. 5.53% 4.22% 4.74% 5.05% 24.72% 23.46%
Table III. Performance of RNGCF w.r.t different lengths of sequential dependencies. Bold scores are the best.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
</tr>
<tr>
<td>0</td>
<td>0.1850</td>
<td>0.1535</td>
<td>0.0670</td>
<td>0.0557</td>
<td>0.0501</td>
<td>0.0386</td>
</tr>
<tr>
<td>1</td>
<td>0.1913</td>
<td>0.1589</td>
<td>0.0688</td>
<td>0.0563</td>
<td>0.0555</td>
<td>0.0423</td>
</tr>
<tr>
<td>2 (default)</td>
<td>0.1944</td>
<td>0.1625</td>
<td>0.0707</td>
<td>0.0582</td>
<td>0.0575</td>
<td>0.0442</td>
</tr>
</tbody>
</table>

Table IV. Performance of RNGCF w.r.t different orders of sequential dependencies. Bold scores are the best.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
</tr>
<tr>
<td>Default</td>
<td>0.1944</td>
<td>0.1625</td>
<td>0.0707</td>
<td>0.0582</td>
<td>0.0575</td>
<td>0.0442</td>
</tr>
<tr>
<td>Reverse</td>
<td>0.1900</td>
<td>0.1589</td>
<td>0.0683</td>
<td>0.0551</td>
<td>0.0513</td>
<td>0.0395</td>
</tr>
<tr>
<td>Rand-1</td>
<td>0.1913</td>
<td>0.1597</td>
<td>0.0681</td>
<td>0.0533</td>
<td>0.0501</td>
<td>0.0388</td>
</tr>
<tr>
<td>Rand-2</td>
<td>0.1918</td>
<td>0.1602</td>
<td>0.0682</td>
<td>0.0553</td>
<td>0.0515</td>
<td>0.0397</td>
</tr>
</tbody>
</table>

C. Impact of Sequential Dependencies

1) Analysis of Embedding Similarity: As we analyze in Section I, the average operation in LightGCN [7] may confuse the orders of high-order paths, which may make the embeddings of nodes excessive similar. To verify this, we define the average cosine similarity of the embeddings of interconnected nodes:

\[
\frac{1}{|R^+|} \sum_{(u,i) \in R^+} \left( \frac{c_u c_i^T}{\|c_u\| \times \|c_i\|} \right),
\]  

where \(R^+\) is a set of the observed interactions. Figure 4 shows the similarity of embeddings learned by three models (MF, LightGCN, and RNGCF). The similarity of embeddings learned by LightGCN is higher than that of MF. This indicates graph convolution makes the embeddings more similar, which is consistent with LightGCN’s finding. The similarity of embeddings learned by RNGCF is lower than that of LightGCN, but the performance of RNGCF does not decrease. This demonstrates that the average operation may make embeddings excessive similar.

2) Impact of Sequential Dependencies on Prediction: To discuss the impacts of sequential dependencies on prediction, we feed different variants of the outputs from different graph convolution layers into GRU. Table III and Table IV show the impacts of the length and the order of the outputs. Note that we sample two kinds of layers into GRU. Table III and Table IV show the impacts of the sequential dependencies.

D. Ablation Studies

We perform ablation studies on RNGCF to show how the components of RNGCF affect performance. Table V shows the results of RNGCF and its variants on three datasets.

- **Remove Dropout**: We conduct RNGCF without Dropout on three datasets. We find that the performance on three datasets is significantly worse. The results show that the dropout can effectively regularize our model to achieve better performance.

- **Remove GRU**: RNGCF without GRU achieves poor results on three datasets. GRU is used to exploit the sequential dependencies in multi-hop paths. The results may indicate the importance of the sequential dependencies.

Table V. Performance of RNGCF and its variants. Bold scores are the best and underlined scores are the second best.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon-Book</th>
<th>Gowalla</th>
<th>Yelp2018</th>
<th>Amazon-Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
<td>Recall</td>
<td>NDCG</td>
</tr>
<tr>
<td>RNGCF</td>
<td>0.1944</td>
<td>0.1628</td>
<td>0.0707</td>
<td>0.0582</td>
<td>0.0570</td>
<td>0.0442</td>
</tr>
<tr>
<td>Remove GRU</td>
<td>0.1850</td>
<td>0.1535</td>
<td>0.0688</td>
<td>0.0565</td>
<td>0.0501</td>
<td>0.0386</td>
</tr>
<tr>
<td>Remove Dropout</td>
<td>0.1841</td>
<td>0.1507</td>
<td>0.0673</td>
<td>0.0544</td>
<td>0.0541</td>
<td>0.0426</td>
</tr>
<tr>
<td>Remove AGU</td>
<td>0.1563</td>
<td>0.1252</td>
<td>0.0526</td>
<td>0.0433</td>
<td>0.0409</td>
<td>0.0323</td>
</tr>
<tr>
<td>Transformer</td>
<td>0.1894</td>
<td>0.1580</td>
<td>0.0690</td>
<td>0.0564</td>
<td>0.0546</td>
<td>0.0423</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.1784</td>
<td>0.1562</td>
<td>0.0689</td>
<td>0.0563</td>
<td>0.0487</td>
<td>0.0377</td>
</tr>
</tbody>
</table>

E. Hyper-parameter Studies

1) Study on dimension of embeddings: We conduct a dimension study on MF, LightGCN, RNGCF, NGAT4rec on Yelp2018, and Amazon-Book. The results of the experiments are shown in Figure 5. As the dimension increases from 16 to 128, the performance of all models increases. The five methods all apply dot product to compute the relevance of items and users. But the limitations of dot product function will become weaker as the dimension increases. RNGCF outperforms all models on all dimensions, which more forcefully indicates RNGCF is effective.

2) Study on number of feature aggregation layers: Table VI shows the performance at different layers (from 1 to 3) and the percentage of relative improvement on each metric. As the number of feature aggregation layers increases from 1 to 3, the performance of all models increases. RNGCF outperforms all models on all dimensions. In particular, RNGCF’s relative improvements over the strongest baselines w.r.t. Recall@20 are the largest when the number of layers reaches 3.

V. CONCLUSION

In this work, we propose a new framework named RNGCF. RNGCF takes a user’s ID and a candidate item’s ID as inputs and apply GCNs to output the user’s preference for the candidate item.
Different from GCN-based previous methods on our task, we design an adaptive unit to adaptively construct the embeddings of users and items based on the sequential dependencies. Extensive experiments on three benchmark datasets show that our model outperforms state-of-the-art models consistently and each component of RNGCF is effective. In particular, RNGCF's relative improvements over the state-of-the-art models consistently and each component of RNGCF is 5.53\%, 4.22\%, 4.74\%, and 24.72\% in Gowalla, Yelp2018, and Amazon-Book, respectively. In future work, we will study how to exploit auxiliary information such as item knowledge graphs, social networks, and multimedia content for our task.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China [No. 62002230], the Natural Science Foundation of Guangdong Province [No. 2019A1515011181], the Science and Technology Innovation Commission of Shenzhen under Grant [No. JCYJ20190808162613130], and the Shenzhen high-level talents program.

REFERENCES


Deep Self-Attention for Sequential Recommendation

Beichuan Zhang, Zhijiao Xiao* and Shenghua Zhong
College of Computer Science and Software Engineering, Shenzhen University, China
zhangbeichuan2019@email.szu.edu.cn, {cindyxzj, csshzhong}@szu.edu.cn

Abstract—Sequential recommendation aims to recommend the next item that a user will likely interact with by capturing the useful sequential patterns from users’ historical behaviors. Recently, it has become an important and popular component in various e-commerce platforms. As a successful network, Transformer has been widely used to adaptively capture the dynamics of users’ historical behaviors for sequential recommendation. In recommender systems, the size of embedding is usually set to be small. Under small embedding, the dot-product in Transformer may have the limitation on calculating the complex relevance between keys and queries. To address the common but neglected issue, in this paper, we present a new model, Deep Self-Attention for Sequential Recommendation (DSASrec), which proposes a chunking deep attention to compute attention weights. The chunking deep attention has two modules: a deep module and a chunking module. The deep module is used to improve the nonlinearity of the attention function. The chunking module is used to calculate attention weights several times like the multi-head attention in Transformer. Extensive experiments on three benchmark datasets show that our model can achieve state-of-the-art results. Our implementation is available in PyTorch.

Keywords—Recommender System, Transformer, Dot-product, Chunking Representation, Deep Learning

I. INTRODUCTION

Recommender system has become an important prevalent component in real-world applications. Learning the embeddings of users and items is an essential topic in recommender systems [1]–[4]. Beyond using the embeddings of users, sequential recommendation considers the sequential patterns in users’ historical behaviors as the pre-existing features of users. To exploit the sequential patterns, Transformer [5] has been widely deployed to sequential recommendation. For example, SASRec [6] tried to capture the dynamics of users’ historical behaviors via Transformer instead of using Recurrent Neural Networks (RNNs). BERT4rec [7] introduced a deep bidirectional sequential self-attention model and a Cloze objective to the field of recommender systems.

Although the previous methods have been proven effective, the previous methods fail to consider the dot-product in Transformer may have the limitation on calculating the complex relevance between keys and queries. In most cases, Transformer is used in the field which has high dimensional embeddings. For example, Transformer paper and GPT [8] paper used 512-dimensional vectors and 768-dimensional vectors to represent words, respectively. In recommender systems, the size of embedding is usually set to be small. For example, SASRec used 50-dimensional vectors to represent items. NCF [9] used 64-dimensional vectors to represent items and users. The size of embedding vector has a great influence on the limitation of dot-product [9]. Suppose we have a user-item interaction graph $G$ as Figure 1(left). There are the similarity relations between $u_1$, $u_2$ and $u_3$ as $s_{23} > s_{12} > s_{13}$, where $s_{ab}$ indicates the similarity of user $a$ and user $b$. When we project the users into 2D space, the geometric relations of $u_1$, $u_2$ and $u_3$ can be expressed by dot-product as in Figure 1(right). There are other similarity relations about $u_4$ as $s_{41} > s_{43} > s_{42}$. However, the relations $s_{41} > s_{43} > s_{42}$ can’t be expressed accurately in the 2D space. If we place $v_4$ closest to $v_1$ as the red vectors in Figure 1, $v_4$ is closer to $v_2$ than $v_3$. It would contradict $s_{41} > s_{43} > s_{42}$. Thus, under small embeddings, using dot-product may lead to that the complex relevance between keys and queries is ignored.

To address the aforementioned problems, in this paper, we present a new model, Deep Self-Attention for Sequential Recommendation (DSASrec). DSASrec takes a user’s historical behaviors and a candidate item as input and outputs the user’s preference for the candidate item. Specifically, we first project users’ historical behaviors into vector representations and then apply a self-attention mechanism to predict users’ preferences. Distinct from existing works [6], [10], we propose a chunking deep attention (CDA) to compute attention weights. The chunking deep attention has two modules: a deep module and a chunking module. The deep module in CDA is used to improve the nonlinearity of attention function. The chunking module in

\[1\] https://github.com/Book1996/DSASrec

DOI reference number: 10.18293/SEKE2021-035

Fig. 1. An example of dot-product’s limitation. Since $u_4$ is most similar to $u_1$, $u_4$ only can be piloted as the red vectors. However, no matter how $u_4$ is piloted, it cannot satisfy this relationship: $s_{41} > s_{43} > s_{42}$. 
CDA is used to calculate attention weights several times like the multi-head attention in Transformer. We perform extensive experiments on four standard large real-world datasets, and the results show our model can achieve state-of-the-art results. To justify the designs in our model, we further conduct ablation studies on DSASrec. The results show each component of DSASrec has contributions to performance. To summarize, this work makes the following main contributions:

- We point out the dot-product in Transformer may have the limitation on calculating the relevance of keys and queries when the size of embedding is set to be small.
- We propose a new model DSASrec, which applies chunking deep attention instead of the multi-head attention in Transformer to model attention weights.
- We demonstrate our proposed can achieve state-of-the-art results by extensive experiments on four standard large real-world datasets.

II. RELATE WORK

A. Attention Mechanism

Attention mechanism can be described as a weighted sum of values, where the weights assigned to each value are computed by a compatibility function. Attention mechanism has become more and more popular in various tasks such as recommender system, machine translation and multimedia. Recently, Transformer was proposed and achieved promising empirical results in machine translation. Due to the efficiency of Transformer, substantial research focuses on improving the performance of Transformer. For example, Transformer-XL introduced a segment-level recurrence mechanism and a novel positional encoding scheme to learn sequential dependency beyond a fixed-length without disrupting temporal coherence. Transformer-XL learned dependency that is 80% longer than RNNs and 450% longer than vanilla Transformers. Reformer replaced dot-product attention by using locality-sensitive hashing and reduced the complexity of dot-product. Synthesizer proved that using dot-product to learn attention weights from token-token (query-key) interactions was useful but not that important.

B. Sequential Recommendation

Most early researches in sequential recommendation use Markov Chains (MCs) to estimate users’ preference for items. FPMC combined Matrix Factorization (MF) and MC for each user owning a personalized transition matrix. Extensive experiments showed FPMC could outperform MF and MCs. Fossil fused similarity-based methods with MC to tackle sparsity issues and the long-tailed distribution of datasets. With progress in deep learning, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) based methods have proliferated. Since the MC-based methods were difficult to consider all users’ historical behaviors, GRU4Rec introduced a ranking loss function and used Gated Recurrent Unit (GRU) to exploit users’ historical behaviors. Caser argued that not all adjacent actions had dependency relationships. Hence, they proposed a CNN-based model that regarded users’ historical behaviors in the latent space as an “image”. Inspired by Transformer, several models were proposed to adaptively capture the heterogeneous, polysemous relationship between items in dynamic sequence for sequential recommendation. For example, SASRec balanced long-term pattern and predictions based on relatively several previous actions via Transformer. SASRec is over ten times faster than RNN and CNN-based methods with GPUs and achieved state-of-the-art results. TiSASRec combined the advantages of absolute position and relative time intervals to learn the weights of different items. BERT4rec employed the deep bidirectional self-attention to model user behavior sequences. SSE-PT introduced additional personalized embeddings to improve the performance of Transformer for sequential recommendation. Although the previous methods have been proven effective, they ignore the limitation of dot-product when the size of embedding vector is small.

III. PROPOSED METHOD

In this section, we present the architecture of DSASrec, which concludes an embedding layer, stacked deep self-attention blocks, and a prediction layer as Figure 2.
A. Problem Formulation

In sequential recommendation, let \( U = \{u_1, u_2, \ldots, u_{|U|}\} \) be a set of users, \( I = \{i_1, i_2, \ldots, i_{|I|}\} \) be a set of items, and \( S^u = \{s^u_1, s^u_2, \ldots, s^u_{|S^u|}\} \) be a historical interaction sequence for a user \( u \in U \), where \( s^u_i \in I \) is the item that \( u \) has interacted with at time step \( t \). Given the interaction history \( S^u \), the sequential recommendation seeks to predict the next item that user \( u \) will interact with.

B. Embedding Layer

Following the prior works \([6], [10]\), we firstly transform the sequence \( S^u = \{s^u_1, s^u_2, \ldots, s^u_{|S^u|}\} \) into a fixed-length sequence \( \{s^u_1, s^u_2, \ldots, s^u_d\} \), where \( N \) is a hyper-parameter meaning maximum sequence length. If the length of \( S^u \) is less than \( N \), we add zero-padding to the left side. If the length of \( S^u \) is greater than \( N \), we only consider the most recent \( N \) interactions. Then, we create an item embedding matrix \( M \in \mathbb{R}^{d \times N} \) and apply a lookup layer on \( M \) to transform \( S^u \) into vector representations, where \( d \) is latent dimensionality. We inject a learnable position embedding \( P \in \mathbb{R}^{N \times d} \). Thus, the input embedding \( E^u \) corresponding with \( S^u \) is defined as:

\[
E^u = \begin{bmatrix}
M_{s^u_1} + P_1 \\
M_{s^u_2} + P_2 \\
\vdots \\
M_{s^u_N} + P_N
\end{bmatrix},
\]

(1)

C. Deep Self-Attention Block

1) Deep Module: An attention function can be described as mapping a query and a set of key-value pairs to a weighted sum of the values. In Transformer, the attention function is defined as:

\[
\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V,
\]

(2)

where \( Q \in \mathbb{R}^{N \times d} \), \( K \in \mathbb{R}^{N \times d} \), \( V \in \mathbb{R}^{N \times d} \), \( N \) is the length of sequence length and \( d \) is latent dimensionality as Section II-B shows. To address the limitation of dot-product, our method replaces the matrix multiplication term \( QK^T \) with a multi-layer perceptron (MLP). The deep attention can be defined as:

\[
\text{Deep Attention}(Q, K, V) = \text{Softmax}(\text{MLP}(QK)) V,
\]

(3)

where \( Q \) is the matrix that includes all key-query pairs and MLP: \( \mathbb{R}^{N \times N \times d} \rightarrow \mathbb{R}^{N \times N} \). The element of \( QK \) is defined as:

\[
QK_{mn} = Q_{m}||K_{n},
\]

(4)

where \( Q_m \) is \( m-th \) row of \( Q \), \( K_n \) is \( n-th \) row of \( K \) and || is the concatenation operation. Recall that we add zero-paddings to the left side, if the length of \( S^u \) is less than \( N \). And our model should consider only the first \( h \) items when predicting the \((h+1)-th\) item. Thus, some key-query pairs are useless and should not be used to compute corresponding weights. Formally, there are three cases that would let \( QK_{mn} \) be an useless key-query pair: Case 1: \( m < n \); Case 2: \( Q_m \) represents pad item; Case 3: \( K_n \) represents pad item. In order to alleviate the computing cost of MLP, we highly optimize our code to no longer use these useless key-query pairs.

2) Chinking Module: Transformer has found it beneficial to linearly project the queries, keys and values several times with different, learned linear projections. We keep the multi-head mechanism to disentangle different information from representation by splitting the queries, keys and values evenly into \( P \) chunks as follows:

\[
K = (K^1; K^2; \ldots; K^P),
\]

\[
Q = (Q^1; Q^2; \ldots; Q^P),
\]

\[
V = (V^1; V^2; \ldots; V^P),
\]

(5)

where \( K^j, Q^j \), and \( V^j \in \mathbb{R}^{N \times d} \). The chinking deep attention (CDA) are defined as:

\[
\text{CDA}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_d),
\]

(6)

where \( \text{head}_j = \text{Deep Attention}(Q^j, K^j, V^j) \).

3) Self-Attention: Recently, a self-attention method was proposed which used the same objects as queries, keys, and values \([5]\). Following previous methods \([6], [10]\), we apply the self-attention mechanism to capture the sequential patterns in users’ historical behaviors. The deep self-attention (DSA) in our case is defined as follows:

\[
\text{DSA}(E^u) = \text{CDA}(E^u, E^u, E^u).
\]

(7)

Following Transformer, we employ two position-wise fully connected feed-forward networks (FFN) with a ReLU activation in between to strengthen the performance of DSA as follows:

\[
A = \text{LayerNorm}(E^u + \text{Dropout}(\text{DSAB}(E^u))),
\]

\[
A' = \text{ReLU}(W_1 + b_1)W_2 + b_2,
\]

\[
Z^u_i = \text{LayerNorm}(A + \text{Dropout}(A')),
\]

(8)

where \( W_1 \in \mathbb{R}^{d' \times d}, b_1 \in \mathbb{R}^{d'}, W_2 \in \mathbb{R}^{d' \times d'}, b_2 \in \mathbb{R}^d \) are model parameters, \( d' \) is a hyper-parameter and \( Z^u_i \in \mathbb{R}^{N \times d} \) indicates the output of the first deep self-attention block. For the sake of simplicity, we define the entire deep self-attention block (DSAB) as follows:

\[
Z^u_i = \text{DSAB}(E^u).
\]

(9)

We stack DSABs to capture more complex feature transition. The \( b-th \) DSAB is defined as:

\[
Z^u_i = \begin{cases}
\text{DSAB}(E^u) & b = 1, \\
\text{DSAB}(Z^u_{i-1}) & b > 1.
\end{cases}
\]

(10)

D. Prediction Layer

After an embedding layer and \( B \) deep self-attention blocks, the behaviors sequence \( S^u \) is transformed into \( Z^u_B \in \mathbb{R}^{N \times d} \). Following the prior works \([6], [10]\), we calculate user \( u \)’s preference for item \( o \in I \) through a dot-product operation as follows:

\[
y^u_{o,j} = z_j M^u_o.
\]

(11)

where \( z_j \) denotes the \( j-th \) row of \( Z^u_B \), \( M_o \) denotes the embedding of item \( o \) and \( y^u_{o,j} \) denotes the possibility of item \( o \) being the \((j+1)-th\) item for user \( u \) given the previous \( j \) items.

E. Model Training

Following previous methods \([6], [23]\), we adopt a binary cross entropy loss to optimize DSASrec, which is defined as:

\[
- \sum_{u \in U} \sum_{1 \leq j < N} \left[ \log(\sigma(y^u_{o,j})) + \sum_{o' \notin S^u} \log(\sigma(1 - y^u_{o',j})) \right],
\]

(12)

where \( \sigma \) is Sigmoid function, \( o' \) indicates a negative item and \( o_j \) indicates an expected item. It is worth noting that when our model inputs a sequence \( \{s^u_1, s^u_2, \ldots, s^u_{|S^u|-1}\} \) and its expected output is...
Table I. The statistics of the datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Items</th>
<th>avg. actions/user</th>
<th>avg. actions/item</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beauty</td>
<td>52,240</td>
<td>57,289</td>
<td>6.9</td>
<td>7.6</td>
<td>0.4M</td>
</tr>
<tr>
<td>Games</td>
<td>31,013</td>
<td>23,715</td>
<td>12.1</td>
<td>9.3</td>
<td>0.3M</td>
</tr>
<tr>
<td>Steam</td>
<td>334,730</td>
<td>13,047</td>
<td>11.0</td>
<td>282.5</td>
<td>3.7M</td>
</tr>
<tr>
<td>ML-1M</td>
<td>6,040</td>
<td>3,416</td>
<td>163.5</td>
<td>289.1</td>
<td>1.0M</td>
</tr>
</tbody>
</table>

For alleviating calculation, as in [6], [23], we randomly generate one negative item \( o' \) for each of expected items.

IV. EXPERIMENTS

A. Experimental Setup

1) Compared Methods: To show the effectiveness of our approach, we compare it with the following state-of-the-art methods:

- **GRU4Rec** [24]. GRU4Rec is an RNN based approach for recommendations, which introduced a novel ranking loss function and GRU in sequential recommendation.
- **Caser** [22]. This method viewed the sequence of recent items as an ‘image’ and used convolutional filters to learn sequential patterns as obtained local features.
- **SASRec** [6]. This method applied Transformer to balance long-term pattern and predictions based on relatively several previous actions.
- **TiSASrec** [10]. This method combined the advantages of absolute position and relative time intervals to learn attention weight.
- **BERT4rec** [7]. This method employed the deep bidirectional self-attention to model user behavior sequences.
- **SSE-PT** [23]. This method introduced additional personalized embeddings to improve the performance of Transformer model for sequential recommendation.

2) Dataset: We evaluate our method on four datasets from three real-world platforms. The four datasets are exactly as same as SASRec used. We request the three datasets from the SASRec. These public datasets have different domains, sizes, and sparsity. In the preprocessing stage, we closely follow the common procedure from SASRec. For all datasets, we treat a review as a padding item and filter out cold-start users and items with fewer than 5 interactions. Each dataset is split into two parts: (1) the most recent action for testing, (2) all remaining actions for training. The statistics of the dataset are shown in Table I.

- **MovieLens**: A widely used benchmark dataset for evaluating collaborative filtering algorithms. We use the version (MovieLens-1M) that includes 1 million user ratings.
- **Amazon**: A series of datasets were introduced in [25]. Followed by the existing work [6], we consider two categories: ‘Beauty’ and ‘Games’.
- **Steam**: A dataset is crawled from Steam by SASRec [6]. It includes rich information like users’ play hours, pricing information, media scores, categories, and developers.

3) Evaluation Metrics: To evaluate recommendation, we use the same protocols as previous methods [6]. Hit@10 and NDCG@10. In the testing phase, for each user, we randomly sample 100 items and rank these items with the most recent action.

Table II. The recommendation results. Bold scores are the best and underlined scores are the second best.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Metrics</th>
<th>Beauty</th>
<th>Games</th>
<th>Steam</th>
<th>ML-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU4Rec+</td>
<td>Hit@10</td>
<td>0.3949</td>
<td>0.6599</td>
<td>0.8018</td>
<td>0.7501</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.2556</td>
<td>0.5282</td>
<td>0.5959</td>
<td>0.5513</td>
</tr>
<tr>
<td>CASER</td>
<td>Hit@10</td>
<td>0.4264</td>
<td>0.5282</td>
<td>0.7874</td>
<td>0.7886</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.2547</td>
<td>0.3214</td>
<td>0.5381</td>
<td>0.5538</td>
</tr>
<tr>
<td>SASrec</td>
<td>Hit@10</td>
<td>0.4852</td>
<td>0.7412</td>
<td>0.8716</td>
<td>0.8132</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3211</td>
<td>0.5633</td>
<td>0.6211</td>
<td>0.5842</td>
</tr>
<tr>
<td>TiSASrec</td>
<td>Hit@10</td>
<td>0.4629</td>
<td>0.7323</td>
<td>0.8657</td>
<td>0.8125</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3016</td>
<td>0.5437</td>
<td>0.6228</td>
<td>0.5711</td>
</tr>
<tr>
<td>BERT4rec</td>
<td>Hit@10</td>
<td>0.4952</td>
<td>0.7499</td>
<td>0.8755</td>
<td>0.8266</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3311</td>
<td>0.5566</td>
<td>0.6315</td>
<td>0.6004</td>
</tr>
<tr>
<td>SSE-PT</td>
<td>Hit@10</td>
<td>0.5028</td>
<td>0.7634</td>
<td>0.8764</td>
<td>0.8288</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3370</td>
<td>0.5622</td>
<td>0.6378</td>
<td>0.6122</td>
</tr>
<tr>
<td>DSASrec</td>
<td>Hit@10</td>
<td>0.5341</td>
<td>0.7826</td>
<td>0.8803</td>
<td>0.8294</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3645</td>
<td>0.5672</td>
<td>0.6416</td>
<td>0.6138</td>
</tr>
</tbody>
</table>

Table III. The computing speed(s) for one epoch. Underlined scores indicates the computation cost of DSASRec is 30% higher than that of SASRec.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Datasets</th>
<th>Beauty</th>
<th>Games</th>
<th>Steam</th>
<th>ML-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>DSASRec</td>
<td>24</td>
<td>13</td>
<td>70</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>SASRec</td>
<td>23</td>
<td>13</td>
<td>69</td>
<td>4</td>
</tr>
<tr>
<td>100</td>
<td>DSASRec</td>
<td>24</td>
<td>17</td>
<td>110</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>SASRec</td>
<td>23</td>
<td>14</td>
<td>91</td>
<td>6</td>
</tr>
<tr>
<td>150</td>
<td>DSASRec</td>
<td>27</td>
<td>19</td>
<td>154</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>SASRec</td>
<td>24</td>
<td>15</td>
<td>120</td>
<td>6</td>
</tr>
<tr>
<td>200</td>
<td>DSASRec</td>
<td>26</td>
<td>22</td>
<td>200</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>SASRec</td>
<td>34</td>
<td>20</td>
<td>150</td>
<td>7</td>
</tr>
</tbody>
</table>

4) Parameter Settings: The proposed DSASRec is implemented on PyTorch, we use two deep self-attention blocks, and each deep self-attention block contains three network layers. The number of neurons of each layer in MLP is 80, 60, 1, respectively. Item embeddings in the embedding layer and prediction layer are shared. The optimizer is the Adam optimizer from [26]. The learning rate is 0.001; batch size is 64; the dropout rate is 0.2 for ML-1m and the Steam, 0.3 for the other datasets. By following the existing work [6], the maximum sequence length \( N \) is set to 200 for ML-1m and 50 for the other three datasets. For all methods, the embedding size \( d \) is searched in \{30, 40, 50, 60\}. The learning rate is searched in \{0.005, 0.001, 0.0005, 0.0001\}. The coefficient \( \lambda \) of L2 regularization term is tuned in \{10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}.

B. Performance Comparison

Table II presents the recommendation performances of state-of-the-art methods on the four datasets regarding Hit@10 and NDCG@10. Table III presents the running time(s) of SASRec and DSASRec for each dataset. The main observations are as follows:

- **DSASRec** achieves the best results on all datasets. In particular, DSASRec significantly performs better on sparse datasets (e.g., Beauty, Game), where DSASRec’s relative improvements over the strongest baselines w.r.t. Hit@10 are 6.22%, 2.51% in Beauty, Games. These results show the high effectiveness of DSASRec.
- **Compared with SASRec**, DSASRec significantly performs better on sparse datasets (e.g., Beauty, Game). This comparison shows that compared with the dot-product operation, deep neural networks may need fewer data to learn transition between items.
On the other hand, the dot-product operation is more likely to overfitting when a dataset is sparse. As discussed in [27], product features are more suitable for memorization, and deep neural networks can generalize better.

- As discussed in [II-C1], we avoid computing useless weights to alleviate the computing cost of MLP, such as the weights for pad items. Thus, the computational complexity of the attention function in our model is \(O(Cd^2/P)\), where \(d\) is the latent dimensionality, \(C\) is the number of valid key-query pairs and \(P\) is the number of chunks. The computational complexity of the attention function in SASrec is \(O(N^2d)\), where \(N\) is the maximum sequence length. Table III shows the computational complexity of attention function in SASrec and DSASrec in practice. The results show that although CDA generates some additional computing costs compared with dot-product, in most cases, the computing cost of DSASrec is not 30\% higher than that of SASrec.

C. Impact of Limitation of Dot-product

Table [IV] shows the results of DSASrec w.r.t different depths of MLP in CDA. Figure 4 shows the variances of attention weights learned by SASrec and DSASrec. The main observations are as follows:

- Compared with the dot-product, MLP can learn more sophisticated features. Thus, the limitation of the dot-product in Transformer may be related to nonlinear ability. To verify this, we conduct DSASrec with different depths of MLP, because MLP with different depths has different nonlinear ability. Table [IV] shows the results of DSASrec w.r.t different depths of MLP. The results show that stacking network layers from 1 to 3 can boost performance. Overfitting issues emerge in Beauty when we stack four network layers. This may demonstrate the representation ability of attention function plays a role in recommendation performance.

- Figure 4 shows the variances of attention weights learned by SASrec are lower than that of DSASrec. These results may indicate the attention weights learned by DSASrec are more decentralized and diversified and CDA can learn more the pluralistic relevance between keys and queries.

D. Ablation Studies

We perform ablation studies on DSASrec to show how the components of DSASrec affect performance. Table [V] shows the results of DSASrec and its variants on three datasets.

- **Remove PE (Positional Embedding):** DSASrec without PE achieves poor results on all datasets. This indicates the order information is important to learn the sequential patterns for sequential recommendation.

- **Remove Dropout:** On all datasets, this variant is significantly worse than the default model. This shows the dropout effectively alleviates overfitting problems in DSASrec.

- **Remove FFN:** We apply FFN to considers relationship between elements in vectors. DSASrec without FFN achieves poor results. Modeling the relationship between elements in vectors can lead to better performance.

E. Scalability of Chunking Deep Attention

Transformer has been widely used to adaptively captures the dynamics of the sequential patterns for sequential recommendation. To show the scalability of CDA, we apply CDA to SASrec, TiSASrec, and BERT4rec. Figure [3] shows SASrec+CDA and TiSASrec+CDA can achieve better performance than SASrec, TiSASrec. But BERT4rec+CDA achieves poor performance than BERT4rec. Compared with BERT4rec, SASrec and TiSASrec have relatively simple structures. The results show CDA may be more suitable for a relatively simple method that uses Transformer.

![Fig. 3. Performance of SASrec, TiSASrec and BERT4rec w.r.t the scalability of CDA on HR@10.](image)

![Fig. 4. The variances of attention weights learned by SASrec and DSASrec.](image)

Table IV. Performance comparison of DSASrec w.r.t number of layers in MLP. Bold scores are the best and underlined scores are the second best.

<table>
<thead>
<tr>
<th>Layer #</th>
<th>Metrics</th>
<th>Beauty</th>
<th>Games</th>
<th>Steam</th>
<th>ML-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 layer</td>
<td>Hit@10</td>
<td>0.5141</td>
<td>0.7608</td>
<td>0.8634</td>
<td>0.8094</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3536</td>
<td>0.5583</td>
<td>0.6204</td>
<td>0.5632</td>
</tr>
<tr>
<td>2 layers</td>
<td>Hit@10</td>
<td>0.5305</td>
<td>0.7649</td>
<td>0.8686</td>
<td>0.8165</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3622</td>
<td>0.5604</td>
<td>0.6252</td>
<td>0.5957</td>
</tr>
<tr>
<td>3 layers</td>
<td>Hit@10</td>
<td>0.5305</td>
<td>0.7734</td>
<td>0.8803</td>
<td>0.8294</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.5341</td>
<td>0.5672</td>
<td>0.6416</td>
<td>0.6138</td>
</tr>
</tbody>
</table>

Table V. Performance of DSASrec and its variants. Bold scores are the best and underlined scores are the second best.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Metric</th>
<th>Beauty</th>
<th>Games</th>
<th>Steam</th>
<th>ML-1M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Hit@10</td>
<td>0.5341</td>
<td>0.7826</td>
<td>0.8803</td>
<td>0.8294</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3645</td>
<td>0.5779</td>
<td>0.6416</td>
<td>0.6138</td>
</tr>
<tr>
<td>Remove PE</td>
<td>Hit@10</td>
<td>0.5205</td>
<td>0.7758</td>
<td>0.8456</td>
<td>0.8115</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3558</td>
<td>0.5669</td>
<td>0.6047</td>
<td>0.5888</td>
</tr>
<tr>
<td>Remove FFN</td>
<td>Hit@10</td>
<td>0.5159</td>
<td>0.7677</td>
<td>0.8551</td>
<td>0.8159</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3382</td>
<td>0.5586</td>
<td>0.5743</td>
<td>0.5850</td>
</tr>
<tr>
<td>Remove Dropout</td>
<td>Hit@10</td>
<td>0.5053</td>
<td>0.7526</td>
<td>0.8635</td>
<td>0.8163</td>
</tr>
<tr>
<td></td>
<td>NDCG@10</td>
<td>0.3416</td>
<td>0.5521</td>
<td>0.6337</td>
<td>0.5875</td>
</tr>
</tbody>
</table>
Fig. 5. Performance of DSASrec w.r.t different numbers of chunks on Beauty and Games.

E. Number of Chunks

To study the influence of chunks number, we vary the number of the chunks of queries, keys, and values in the range of \{1, 2, 4, 8\} and show the performance on Beauty and Games datasets in Figure 5 (results on other dataset show similar trends which are omitted for space). Increasing the number of chunks from 1 to 2 leads to better performance. However, the recommendation performance drops when the chunk number increases from 2 to 8. This suggests that the DSASrec suffers from too fine-grained chunks.

V. Conclusions

In this paper, we propose a novel self-attention based method for Sequential Recommendation named DSASrec. It proposes a chunking deep attention (CDA) to computing the attention weights. CDA is used to alleviate the limitation of dot-product in Transformer. Experimental results on four real-world datasets show that DSASrec outperforms state-of-the-art techniques and each component of DSASrec has contributions to performance. In section IV-C we demonstrate the performance of attention-based models can be improved by enhancing the nonlinearity of attention function. The variances of attention weights learned by dot-product attention are lower than the attention weights learned by CDA. In the future, we plan to extend the model by incorporating auxiliary information (e.g., action types, item knowledge, etc.).

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China [No. 62022030], the Natural Science Foundation of Guangdong Province [No. 2019A1515011181], the Science and Technology Innovation Commission of Shenzhen under Grant [No. JCYJ20190808162613130], and the Shenzhen high-level talents program.

REFERENCES

A recommender system to assist conceptual modeling with UML

Maxime Savary-Leblanc*,**, Xavier Le Pallec* Sébastien Gérard**

*Univ. Lille, UMR 9189 CRIStAL, F-59000 Lille, France **CEA LIST, Palaiseau, France

E-mail: maxime.savary-leblanc@univ-lille.fr

Abstract

This paper explores the understudied field of conceptual modeling assistance. More specifically, we focused on the design and application of recommender systems as software assistants for conceptual modeling. Prior work on such systems has shown that trust plays a key role in the acceptance and exploitation of such systems. Consequently, as a starting point of our research, we applied established methods for constructing multi-criteria recommender systems (MCRS) to conceptual modeling in a way which could foster the emergence of trust. Finally, we chose supervised-learning techniques to refine and customize the recommendations generated by these systems. To help us determine the feasibility and practicality of our approach, we designed and implemented a prototype system that assists conceptual modeling with UML. Our system currently recommends class attributes when constructing UML class diagrams. A preliminary evaluation of this tool indicated a strong match between the recommendations provided by our system and personal choices made by the participants.

1 Introduction

Domain knowledge is a prerequisite to produce software design and implementation tailored to stakeholders’ requirements. One common way to formalize that knowledge is achieved through conceptual models, which are commonly used to describe or simulate a system. Acquiring such expertise requires to discuss with knowledgeable stakeholders and/or to get an access to useful documents, which both might not always be easily accessible.

In the same time, more and more model samples can be gathered from multiple sources, what represents an increasing number of already formalized and accessible knowledge pieces. For example, some companies keep archives of internal model repositories [1]. There also exist numerous open source projects that contain models [2] while some modeling tools even offer the possibility to create public projects that are free to browse. However, when it comes to hundreds of thousands models, the time spent browsing them manually and filtering the relevant ones seems unreasonable given the original problem.

One facility for exploiting this large amount of data is to build recommender systems [3] whose recommendations are based on such data. In our work, we decided to investigate the use of recommender systems to support the conceptual modeling activity. Our general research objective is to develop a comprehensive general methodology for designing recommender systems for conceptual modeling. As an initial step, we first developed a working prototype that can serve as a testbed to help in evaluating and refining our approach. The specific objective of this phase of our work was to develop a Multi-Criteria Recommender System (MCRS) that generates semantically meaningful attribute recommendations for UML classes. We used Roy’s method to develop our approach and then performed an initial empirical evaluation as described in Section 5.

To the best of our knowledge, this is the first semantic recommender system for UML class modeling. As such, we feel that it is an important contribution to the modeling community. It provides a means of assisting the modeling process – something that could particularly benefit less experienced modelers. This can, in turn, lead to a broader adoption of model-based software engineering by software developers. In addition and in line with our overall objective, we believe that the approach applied here can be extended to cover other types of recommender systems to be used in modeling.

The remainder of the paper is organized as follows. In Section 2, we explain our approach, while in Section 3, we identify and formalize suitable recommendation criteria. A criteria aggregation method is described in Section 4. Section 5 presents the prototype tool and an initial experiment used to evaluate its suitability and effectiveness. Related work is covered in Section 6. Finally, Section 7 provides a brief summary followed by a discussion of potential future work.

2 Methodology

Most recommender systems implement one of the following three common recommendation methods: collaborative filtering [4], content-based techniques [5], or knowledge-based techniques [6]. Each has advantages and limitations. Common limitations include overspecialization, cold-start is-
3 Criteria formalisation

In this section, we describe the criteria and their rationale, on which each recommendation is assessed.

3.1 In-class recurrence criterion (C1)

In the following, the term ‘owner class’ of an attribute refers to the class that directly owns that attribute.

Rationale. The attribute is often present in classes with the same name as the owner class.

Selection filter. Attributes owned by classes with the same name as RC.

Rating approach. The most frequently occurring candidate gets the highest score and the least frequently occurring candidate gets the lowest non-null score. Candidates that do not appear in a class with the same name as the owner class get a null score.

3.2 In-class exclusivity criterion (C2)

Rationale. The attribute is only present in classes with the same name as the owner class.

Selection filter. Attributes owned by classes with the same name as the owner class.

Rating approach. Candidates which only appear in classes with the same name as the owner class get the highest score. Those that appear equally in all classes of the data set get the lowest score. Candidates that never appear in a class with the same name as the owner class get a null score.

3.3 Attribute synergy criterion (C3)

Rationale. The attribute often describes a class named the same as the owner class in similar models.

Selection filter. Attributes connected to attributes owned by RC through their presence in a common class. Common classes are those classes that share the same name.

Rating approach. The more often a candidate and an attribute of RC appear together in a class, the higher the score. The more that a candidate appears together with different attributes of RC, the higher the score. Candidates which never appear together in a class get a null score.

3.4 Context similarity criterion (C4)

Rationale. The attribute often describes a class named the same as the owner class in similar models.

Selection filter. Attributes owned by classes named |RC| in models which share at least two common classes with M(RC).

Rating approach. Candidates from models that share the highest number of classes with M(RC) get the highest score. Candidates from models which have no class in common get a null score.

4 Utility Function

The third step of our approach consists in constructing a utility function that aggregates the score of each single criterion into a global score, on which to base the ranking. In defining what makes a good explanation in recommender systems, [10] argue that “justifying [a] recommendation is just half of the solution, the second half is to make it scrutable”.

To that end, in this section we first select an aggregation method that enhances system transparency. Then we emphasize support for context adaptability, and, finally, propose a determination process that allows system control through scrutability.

4.1 Utility Function selection

Adomavicius and Kwon [8] identify two major techniques for dealing with multi-criteria ratings to produce an overall rating: heuristic-based and model-based techniques. Heuristic-based techniques compute the score of each item for a given user, based on data derived from observing one specific user, using some heuristic assumption. To perform matching operations, these techniques often require specific knowledge about multiple users, based on their profile and
from collaborative filtering. In contrast, model-based techniques generate a predictive model, typically using statistical or machine-learning methods that best explain the observed data. Once the model becomes available, they use it to estimate the score of individual recommendations.

In our case, the lack of data about the profiles of all users rules out heuristic-based techniques. On the other hand, model-based techniques using machine-learning methods enable the system to learn directly from the user, resulting in finely-tuned data. Consequently, we take a machine-learning model-based approach to determine the overall utility function. Note that, for greater system transparency, the aggregation process must be explainable. Therefore, rather than relying exclusively on machine-learning processes, which are rarely fully explainable, we define the utility function as a weighted sum of criteria rating functions.

We define this function as follows:

Let \( (a, b, c, d) \in [0; 1]^4 \) where \( a + b + c + d = 1 \),

\[
\text{overall}_{RC} : |A| \rightarrow [0; 1] \\
|p| \mapsto a \times s_1 + b \times s_2 + c \times s_3 + d \times s_4
\]

with \( s_1, s_2, s_3, s_4 \) the scores for criteria 1 to 4

The machine-learning process is used to determine the values of the weights \( a, b, c, \) and \( d \).

### 4.2 Context adaptability

Adomavicius and Kwon [11] also note that the aggregation function can have different scopes: total (i.e., when a single aggregation function is learned based on the entire data set), user-based, or item-based (i.e., when a separate aggregation function is learned for each user or item).

In the context of recommending UML attributes for classes, we identify four different possible Contexts i.e. situations. The system will provide recommendations for the following:

- **Context 1**: A class owning no attributes and no other classes in the model.
- **Context 2**: A class owning one or more attributes and no other classes in the model.
- **Context 3**: A class owning no attributes in a model but containing one or more other classes.
- **Context 4**: A class owning one or more attributes in a model and also containing one or more other classes.

Each of the above contexts has access to different information so that not all of the criteria can be applied equally to all of them. For instance, the context similarity criterion C4 relies on the presence of other classes in the model and is, therefore, not applicable to contexts 1 and 2. Consequently, we define the overall utility function in context \( k \) \( \text{overall}_{k,RC} \) as:

\[
\text{overall}_{k,RC}(|p|) = a_k \times s_1 + b_k \times s_2 + c_k \times s_3 + d_k \times s_4 \tag{2}
\]

This results in four different utility functions corresponding to the four different contexts. They are determined individually in the course of the machine-learning process.

### 4.3 Utility function determination

The quality of a recommender system depends primarily on its ability to propose items that the user is likely to choose rather than items the user is unlikely to choose. Therefore, a high-quality recommender system must fit user preferences. Our system offers the possibility to reflect these preferences by assigning values to the weights of the four overall utility functions. This can be done manually, but finding suitable values would likely lead to suboptimal results. Instead, we chose a machine-learning approach to automatically determine these weights.

We collect labelled data through a dedicated interface (presented in section 5.1, and in the web page) during a preference-elicitation phase. This interface first presents multiple situations (a class diagram with recommendation target class) one at a time. A list of unranked candidates —potential recommendations— is displayed for each situation. Using this interface, the user is asked to remove all attributes that do not fit semantically in the presented situation. Once this is completed, the user is then asked to create a ranked list of the top 10 best recommendations from the displayed elements. This task should be repeated for multiple situations in different contexts a sufficient number of times in order to collect enough information to determine the four utility functions. Once this data is collected, it is used to calculate \( a_k, b_k, c_k, d_k \) weights in such a way that they maximize the Mean Top Average Precision metric defined in Section 5.2.2.

### 5 Implementation and Evaluation

In this section we first describe the implementation of our solution followed by a description of the initial evaluation and its results. More details about the implementation and its behaviour are available online1.

#### 5.1 Implementation

**Recommender system.** Our implementation conforms to the standard three-tier architecture pattern2: the data tier, the application tier, and the presentation tier. The data tier consists of a Neo4j3 server which holds the full models data set represented as a graph. The application tier is a Spring Boot4 server exposed as an API which is responsible for computing

1. https://hufamo.univ-lille.fr/modeling-assistant
2. https://www.tandfonline.com/doi/abs/10.1080/10580539608906981
3. https://www.neo4j.com
the scores and the output of the utility function, to produce recommendations. Finally, the presentation tier is a Papyrus\(^3\) plugin which presents recommendations to the user and make it exploitable.

**Supervised Learning Platform.** The aim of the supervised learning approach is to determine a combination of a, b, c, d values that maximizes the accuracy of the system for a specific context. Initially, we start the learning process with equal values for these weights. These values are then varied using a predefined increment while maintaining the constraint specified by equation 1. For each configuration of a, b, c, and d, we calculate the chosen evaluation metric for the system and compare it to the previous maximum value. The highest value is stored as well as the associated configuration of weights. After all configurations are analysed, the one that maximized the chosen metric is selected. This algorithm is coded in Java as part of the application tier. To compute accuracy, the algorithm exploits the labelled data created through our dedicated interface. The labelling interface is a web application coded in HTML/CSS/JS that enables labelled data collection. It takes JSON files and class diagram pictures as input and outputs JSON files containing user preferences as output.

### 5.2 The evaluation

The preliminary evaluation of our approach is based on assessing improvements in the quality of the recommendations, as well as the adequacy of system control, information transparency, and system transparency. To the best of our knowledge, no similar approach can be found in the literature. A replication package contained the labelled data, the original files and the metrics source code is available online\(^6\).

#### 5.2.1 Data gathering

The evaluation involved data from over 95'000 models. These contained approximately 634'000 classes and 616'000 attributes. The models were retrieved from the GenMyModel\(^7\) public repositories by courtesy of Axellience. For quality purposes, we only selected models greater than a minimum size (over 10 kilobytes).

We gathered labelled data according to the method described in Section 4.3. From this, we obtained 9,858 labelled attributes from 30 participants: 9 senior and 4 junior researchers, 3 senior and 12 junior developers from industry, and 2 M.Sc. students. Prior to starting the labelling exercise, participants were asked to answer questions about their familiarity with UML and the extent of their modeling work. On average, participants estimated their knowledge of UML class diagrams to range between fair and good (mean: 3.5 on 5-point Likert scale, std. deviation: 1.0). This assured us that participants had a relatively good understanding of the context and consequently, that the information gathered was semantically meaningful. Participants were asked to respond to up to 20 examples of different situations: 5 per context. In order to minimize the impact of participant fatigue on the results, the 20 examples were randomly displayed and participants were allowed to respond in several sessions.

#### 5.2.2 Evaluation metrics

To more accurately evaluate the attribute recommendations, we compared the ranked results of our system with the ranked preferences as chosen by the users who created their top-5 ranked list. Consequently, we computed metrics for just the top-5 recommended attributes; i.e., the five attributes with the highest scores.

**Precision@5** (P@5) is the proportion of recommended items that a user deemed as belonging in the top-5 list of relevant attributes. In our case, relevant attributes were those that were not excluded by the user from the candidate list.

\[
P@5(\text{set}) = \frac{n^o\text{ of relevant items in system top-5}}{5}\] \tag{3}

**TopPrecision@5** is the proportion of recommended items in the top-5 list provided by the recommender system that are also included in the top-5 set chosen by the user.

\[
TP@5(\text{set}) = \frac{n^o\text{ of common items in user and system top-5}}{5}\] \tag{4}

**TopAveragePrecision@5** (TAP@5) takes ranking into consideration in evaluating the mean average precision of the top-5 of the system. Mean Average Precision (MAP) is a popular metric for measuring recommendation algorithms in information retrieval. We defined TAP@5 as follows:

\[
TAP@5(\text{set}) = \sum_{n=1}^{5} \frac{P(n) \times \text{pos}(n)}{R}\] \tag{5}

where \text{pos}(k) indicates whether the element from system top-5 in position \(k\) matches the position of the element in a user’s top-5 list, while \(R\) refers to the number of elements for which \text{pos}(k) = 1; \(P(k)\) is the ratio of correctly recommended elements over top-k recommended elements.

These metrics can be computed for each ranked set of attributes. Therefore, as users provided several sets of attributes, we considered the means of these metrics as follows:

\[
Mm(S) = \sum_{s \in S} \frac{m(s)}{N}\] \tag{6}

where \(m\) is the metric for which the mean is calculated (i.e., MP, MTP, and MTAP); \(S\) is the data set for which the mean was computed, and \(N\) is the number of elements in \(S\).
Table 1. Labelled data distribution

<table>
<thead>
<tr>
<th>Context</th>
<th>Training</th>
<th>Testing</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>General</td>
<td>Ctx. 1</td>
<td>Ctx. 2</td>
</tr>
<tr>
<td></td>
<td>(205 sets)</td>
<td>(50 sets)</td>
<td>(43 sets)</td>
</tr>
<tr>
<td>Training</td>
<td>8,146</td>
<td>2,462</td>
<td>2,338</td>
</tr>
<tr>
<td></td>
<td>(245 sets)</td>
<td>(61 sets)</td>
<td>(53 sets)</td>
</tr>
<tr>
<td>Testing</td>
<td>1,712</td>
<td>544</td>
<td>562</td>
</tr>
<tr>
<td></td>
<td>(40 sets)</td>
<td>(11 sets)</td>
<td>(9 sets)</td>
</tr>
<tr>
<td>Total</td>
<td>9,858</td>
<td>3,009</td>
<td>2,900</td>
</tr>
<tr>
<td></td>
<td>(245 sets)</td>
<td>(61 sets)</td>
<td>(53 sets)</td>
</tr>
</tbody>
</table>

Table 2. Learned overall functions

<table>
<thead>
<tr>
<th>Context</th>
<th>Overall Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context 1</td>
<td>$0.80 \times s1 + 0.20 \times s2$</td>
</tr>
<tr>
<td>Context 2</td>
<td>$0.03 \times s1 + 0.01 \times s2 + 0.96 \times s3$</td>
</tr>
<tr>
<td>Context 3</td>
<td>$0.51 \times s1 + 0.03 \times s2 + 0.46 \times s4$</td>
</tr>
<tr>
<td>Context 4</td>
<td>$0.56 \times s1 + 0.03 \times s2 + 0.29 \times s3 + 0.12 \times s4$</td>
</tr>
</tbody>
</table>

5.2.3 Metrics results

We obtained 245 sets of labelled data from users, which constitute a corpus of 9,858 attributes distributed for training and testing phases, as presented in Table 1. We trained our overall rating functions with 81% (205 sets) of the total labelled data set and obtained the functions presented in Table 2. The goal of the training was set to the maximization of the MTP@5 metric, as it is the most representative possible improvement of our system when compared to unassisted user selections.

The added value of using machine-learning is demonstrated by the evolution of the metrics before and after the machine-learning process. Both situations only differ in the values and distribution of weights in the utility functions. We set up the initial configuration (i.e. before ML) by setting the weight values to be equal. For instance, we define the initial aggregation function for Context 4 as $\text{overall}_4 = 0.25 \times s1 + 0.25 \times s2 + 0.25 \times s3 + 0.25 \times s4$. The final configuration corresponds to the application of the utility functions defined in Table 2.

We evaluated the overall utility functions on a specific testing data set, which represents 19% of the full labelled data set, (see Table 1). The other 81% were used for training purposes. The results of the metrics evaluation are presented in Table 3.

5.3 Discussion

The initial general MP@5 is pretty high (87.0%). The low impact of the learning process on this score (+4.5%) indicates that the different criteria already strongly converge to recommend user-relevant attributes, and that the impact of the weights on the overall rating functions are, in that case, secondary. The impact of the supervised-learning process becomes more important according to the desired quality of the recommendations. Indeed, the initial low value of MTP@5 increases from 34.7% to 51.0% after the learning process. This means that, on average, more than two attributes of the 5 first recommendations of the system are attributes that participants included in their top-5 best recommendations. Following the learning step, MTP@5 shows the most significant increase among all metrics (+16.3%). The utility functions were defined so as to optimize this metric. MTAP@5 takes differences in recommendations ranking between system and user top-5 into account. Only high-quality recommendations increase this metric, which explains why it has the lowest initial values for all contexts. With a final value of 42.5%, MTAP@5 indicates that, on average, more than 2 attributes of the 5 first recommendations are in participants’ top-5, likely to be in top positions and ordered as the participants expected.

5.4 Evaluation results

The empirical results obtained indicate that our approach provides acceptable results (on average, more than 4 recommended attributes out of 5 are deemed relevant, and also 2 recommendations out of 5 appear in users’ top-5 rankings). However, as pointed out earlier, it is too early to make any firm conclusions about the effectiveness of our approach compared to alternatives until further evaluations are performed. In addition, we can draw the following conclusions from the evaluation:

- the initial effectiveness measure that we proposed here looks as if it could serve as a common metric for future related work.
- The defined criteria do seem to reflect information trustworthiness. Moreover, the rationale behind them can be easily explained, which means that they do support information transparency.
- The linear utility function approach we used allows any overall score to be traced to each criterion used to derive it. This enables users to understand the inner mechanisms of the system and thus supports system transparency.

- The utility function can either be set manually or defined using supervised-learning. These settings allow users to have control over the results that are presented giving them control of the system.

6 Related work

In this section, we review published work in the following related areas: (i) tools that help with semantics-related issues involved in modeling, and (ii) recommender systems for software engineering.
6.1 Conceptual modeling assistance

Although a lot of work has been done on supporting software engineering with software assistants, not much of it has been applied to modeling. Segura et al. [12] recognize the need for assistance during modeling activities and introduce Extremo, an Eclipse plugin for modeling. They propose a framework for integrating diverse data sources into the Eclipse modeling environment. However, the data sources, such as model repositories, must be provided by the user. Koschmider et al. [13] propose a recommendation-based editor for business process modeling. Their system provides users with recommendations about partial process models.

Kogel [14] describes the early stages of a work on modeling recommendations and proposes a prototype providing unranked recommendations. Elkamel et al. [15] present a UML class recommender system that recommends new classes for a UML model. This system measures the similarity between current model classes and existing ones from a repository to recommend the closest matches. In a similar fashion, Cerqueira et al. [16] proposed a content-based approach for recommending UML sequence diagrams.

The above papers highlight both the novelty of and the need for semantic assistance in modeling activities. Surprisingly, while class diagrams remain among the most widely exploited UML classes [17], almost no effort has been conducted to address support for their design.

6.2 Recommender systems for software engineering

Only a few recommendation systems have been applied to modeling to date. In fact, Dyke et al. [18] identify recommender systems as promising new area of research, since recommender systems have already found their way into general software engineering [19] and many of the software lifecycle processes, as defined in ISO/IEC 12207 [20].

Multiple works [21] [22] investigated the application of recommender systems to the field of requirements engineering. Sharma and Sodhi proposed a recommender system [22] to help in dealing with the manual effort required to identify and analyse relevant architectural patterns in the context of a particular set of software requirements. A variety of recommender systems focus on the software construction process, from providing code examples to suggesting modifications [19]. The work described in [23] provide developers with recommendations about API usages and parameters. In [24], Allamanis et al. propose an algorithm which suggests meaningful class and method names to enhance software quality. Some works also involve recommenders for finding relevant answers to developer’s technical questions [25].

All of the above clearly identifies a gap in support for recommender systems in semantic-based assistance for conceptual modeling.

7 Conclusion and Future Work

In this paper, we are seeking to support the conceptual modeling task. As an initial step we proposed, implemented, and evaluated a modeling recommender system. The initial evaluation, involving both practitioners and students and a prototype implemented with Papyrus, indicates that the approach holds promise. A replication package was provided to serve as first comparison point for future works in the domain.

Our plan is to generalize the approach realized in the prototype for other types of models, such as activity and sequence diagrams, to move towards our greater objective of a generic framework for building design-assisting recommender systems. We also plan to conduct further work specifically focusing human-centric aspects. This includes ways to provide users with explanations about recommendations, but also the types of interaction recommender systems should propose to best fit users’ mental design process.

References


Evaluating Visual Explanation of Bug Report Assignment Recommendations

Shayla Azad Bhyan and John Anvik
Department of Mathematics and Computer Science
University of Lethbridge, Alberta, CANADA
E-mail: [s.bhuyan, john.anvik]@uleth.ca

Abstract

Software development projects typically use an issue tracking system where the project members and users can either report faults or request additional features. Each of these reports needs to be triaged to determine such things as the priority of the report or which developers should be assigned to resolve the report. To assist a triager with report assigning, an assignment recommender has been suggested as a means of improving the process. However, proposed assignment recommenders typically present a list of developer names without an explanation of the rationale. This work presents the results of a small user study to validate our approach to visually explaining bug report assignments.

1. Introduction

As the need for global and distributed software projects grows, so does the need for finding people with the required expertise for a given task. Recommendation systems have been proposed as a means for improving the achievement of this goal [1–4]. The typical recommendation system provides a textual list of recommendations with no explanation for why each recommendation was made. As described by Herlocker et al. [5], most current recommendation systems are a black box where transparency is not ensured.

Providing transparency by incorporating the reasoning and data behind a recommendation is an important feature of an effective recommendation system [1, 6], as recent work in “Explainable Artificial Intelligence" shows [7, 8]. Effective visualizations can help to provide this transparency for a recommender system that uses multidimensional data and improve user acceptance rates for recommendations. Making efficient visualization of the recommendations can show how different dimensions were applied in making a recommendation to improve transparency [6], as well as improving a user’s acceptance rate of recommendations [1]. Trintarev et al. [9] surveyed a group of movie-goers and found that the explanations behind the recommendations are as important to users as the recommendations.

Bug report triage recommenders are an example of such a recommender system in software engineering. Bug report triage is the process where a project member, typically a project manager, decides what to do with a bug report. When projects receive many bug reports every day, bug report triage becomes a significant software maintenance issue [10–12]. Also, bug report triage is a tedious task that often shifts development resources away from improving a product to instead managing the project. Within the area of bug report triage recommenders, assignment recommenders are the most commonly researched (e.g. [4, 10, 13]). Typically, proposed assignment recommenders provide a textual list of recommended developers’ names (e.g. [10, 14, 15]). Despite years of research, assignment recommenders have yet to be meaningfully integrated into products such as Bugzilla, GitHub and Jira. Prior studies (e.g. [16]) and informal discussions with developers indicate that one of the barriers to the adoption of such a system is the lack of explanation, leading developers to question and perhaps not trust the recommendations. It is these discussions that motivated this work.

This paper presents an evaluation of our initial work towards providing transparency for bug report triage assignment recommendations using visual explanations. We explore the use of stacked horizontal bars,

---

1We use the term "bug report" to refer to items in a project’s issue tracking system.
a pie chart, and a data table. To assess the impact of the use of these visualizations, we conducted a small user study.

To the best of our knowledge, this area of bug report assignment recommenders has not been explored in the literature. We believe that part of the reason for this is that these recommenders are created typically using machine learning algorithms that make it hard to provide explanations. For example, the two most commonly used algorithms are Support Vector Machines (SVM) [17, 18] and the Random Forest algorithm [19, 20] where determining the rationale for the recommendations is near impossible. Instead, we focused on the use of Multinomial Na"ive Bayes and Topic Modelling, both of which use probabilistic models. The use of a probabilistic model makes for an easier determination of recommendation rationale. In contrast to Multinomial Na"ive Bayes which has been commonly used in the past, the use of Topic Modeling in this area is relatively new.

2. Visualization of Assignment Recommendations

To provide an assignment recommendation for a new bug report, first, the report is turned into a vector of features. Next, the features of the new bug report are given to the trained classifier. In the case of Multinomial Na"ive Bayes, for each potential developer, the set of features that are common between their instances and the new bug report are collected. Then the sum of the conditional probability of each of these features is determined to represent the expertise score of that developer for that bug report. Finally, developers are ranked based on expertise scores. In most cases, bug reports have a lot of relevant words in common. If all of these words were displayed, the graphs would have too much information and be hard to read and understand. Therefore, we chose to display only the most relevant terms based on the TF-IDF score. We empirically found that providing more than five terms did not significantly improve the accuracy.

When choosing the visualizations to explore, we focused on simplicity and familiarity to potential users. Therefore, we chose the data-table format, stacked bar chart and pie chart, as they are commonly used in a variety of applications and are familiar to a wide range of people. Also, we sought visualizations that would allow for the display of information about relative contributions. In our case, that means the individual probabilities of terms or the dominant topic in a bug report will have towards the ranking of developers. Finally, these forms of visualization have been previously used in similar contexts [1, 21, 22].

Figure 1 shows these three types of visual representation for an assignment recommendation using Multinomial Na"ive Bayes. The pie chart presents the important features from each report based on their conditional probability values. If a user clicks on the pie, a new web page opens. This new page shows a pie for each developer and each pie shows the overall conditional probability values for the corresponding recommended developer. The data table shows these same values for each important feature for each developer. The stacked bar chart however shows developers horizontally where each feature is represented by a different colour. The developer who has the highest sum of conditional probability values for all of the selected features is shown at the bottom.2

For the Topic Modelling classifier, the cluster with the shortest distance to the new bug report is determined and the ranked list of developers for that cluster forms the recommendation list. The pie chart shows the recommended developer names and their solved bug report rate for a specific topic. The data table gives the list of developer names with the exact number of reports that the developer solved related to that topic. The stacked bar chart also shows the developer names horizontally with their score. The colour of the bar is related to the selected topic.

3. Evaluation

Our empirical study3 sought answers to three research questions. First, do developers find visual explanations of assignment recommendations easy to understand? Second, do developers trust visual explanations of assignment recommendations? Lastly, which of the three investigated visualizations is preferred?

The web application used in our study consisted of two parts: a web browser plug-in and a web service. To present a subject with visual explanations for the assignment recommendations, we created a web browser plug-in for Google Chrome 4.

To use the plug-in, first, a user opens a bug report in the web browser from a Bugzilla server. We configured the plug-in to only work with bug reports from Mozilla projects (i.e. those with the URL https://bugzilla.mozilla.org), as that was our chosen dataset. Next, the user clicks on a button

2That the top recommendation is shown at the bottom is a result of the graphics library used, not an intentional choice.

3An analytical evaluation of the underlying recommenders was conducted before the study. See [23] for these details.

4https://chrome.google.com/webstore/detail/recommend-expertise/clpdpdhhohhfcnkkipopaeikbngid
labelled “Recommend Experts” in the plug-in in the browser. This makes a request to the web service with the bug report’s id and opens a new browser window containing the response from the web service - an HTML page showing the assignment recommendations in a visual form. Figure 1 shows one of the four visualization web pages that are returned by the web service.\(^5\)

When given the bug report id, the web service queries the issue tracking system for the title and description of the requested report. Stop words are removed and stemming applied to the text before being passed to a classifier. The results from the classifier are then used to create the visualizations. As previously mentioned, only the top five (5) recommended developers are shown to avoid information overload.

### 3.1. User Study

The user study consisted of a within-subject study where all participants received treatment. Our user study\(^6\) consisted of three parts: a demographic survey, presentation of the visualizations with an accompanying survey, and a post-usage survey.

The demographic and post-usage survey was conducted using Qualtrics, and the visualization survey integrated into the web pages was generated by the web service. Participants were asked to complete the demographic survey first, then install the browser plug-in and go through the list of bug reports, and then complete the post-usage survey.

To recruit participants for our study we posted on Reddit in channels like r/learnmachinelearning and r/AskComputerScience. The criteria for participation was to either be in a two-year computer science post-graduate degree (i.e. in an M.Sc.-like program) or have more than one year of software development experience. Interested participants were asked to contact the primary researcher for a study id and further instructions. We were able to recruit fourteen participants.

As previously noted, this research direction is new in the software engineering area. Participants could have been recruited from the Bugzilla project (i.e. the data set used for training the assignment recommender), but we chose to conduct a small study first to assess the viability of our approach before approaching specific project developers. In other words, the purpose of the user study was to gain a general understanding of the effectiveness of visually representing bug report assignment recommendations. By having participants

\(^5\)Examples of the other visualizations can be found in [23].

\(^6\)The study was reviewed by the University of Lethbridge Ethics Committee and assigned protocol number #2019-070.

that were not associated with the particular project for which the assignment recommender was created, we sought to determine a base case for future investigations in this area.

To assess the effectiveness of the visualizations, each participant was given the same set of fifteen (15) links to pre-selected bug reports for the Bugzilla software product. The selected bug reports were randomly chosen from those that had a status of Open (i.e. not Resolved). This was done so that the reports reflected the general level of difficulty of reports present in the issue tracking system for the product (i.e. no consideration was given for the complexity of the bug report in their selection) and so that participants were not biased towards the recommendations by examining “the correct answer” of who should have been recommended as the assignee.

After clicking on a link for a bug report, the participant was taken to the actual bug report in the Mozilla project’s issue tracking system. The participant would then click “Recommend Experts” in the plug-in and the web service would provide the recommendations as part of one of four randomly selected web pages. The participant would also be asked one of two sets of questions depending on the presented visualizations.

The intent of two of the web pages was to present participants with a single type of visualization (stacked bar or pie chart) with data from each of the recommenders. In this way, we could determine if participants preferred the use of one visualization approach over another. The intent of the other two web pages was to determine if participants preferred a particular type of classifier.

For the web pages that presented results from the two different classifiers (Multinomial Naïve Bayes and Topic Modelling), participants were asked several questions: Did they think the visualizations increased their understanding of the recommendation? Did they trust the recommendations? Did they think the visualizations provided enough information? If not, what visualization did they think was missing?

For the web pages where the results from the same classifier were presented, but the visualization differed (i.e. bar vs. pie vs. table), the participants were asked similar questions as before. Did they trust the recommendations? Did they think the visualizations provided enough information? If not, what visualization did they think was missing?

After participants finished using the browser plug-in on the fifteen bug reports, or however many they chose to do, they were asked to complete the post-usage survey. This survey asked their thoughts about our approach to providing visual explanations of bug report
assignment recommendations. Examples of questions asked included: How important is the visual explanation of the recommendations to you? How would you improve the explanation of the recommendations? Did you think that one visualization was enough? Which combination of visualizations would you want for explaining an assignment recommendation?

4. Results

We found that most participants took an hour to complete the study, although one participant took much longer (2.5 hours) 7.

The occupations reported by the participants were: student (3), quality analyst (3), application developer (5), project manager (1), and application architect (1). The participants identified as 64% male and 36% female, and just over half (57%) of the participants had a graduate degree (Masters or Ph.D.). Participants’ development experience varied from less than three years (1), four to nine years (7), and more than nine years (6). Most of the participants (71%) reported having logged a bug report, which indicates that most of them had some form of first-hand knowledge of how bug report assignment works. When asked about their level of familiarity with machine learning, two (2) reported themselves as beginners, and the rest considering themselves to have advanced knowledge.

4.1. Visualisation of Assignment Recommendations

Table 1 shows the results for the questions where we were trying to determine if there was a preference for one visualization over another. We can see that there was a slight preference for the stacked bar chart over the pie chart. We can also see that more than 70% of participants felt that these visualizations provided enough information. There was no notable difference in the preferred visualizations for developers with different experience levels. This may be a result of the participants not being intimately familiar with the project. Note that no participants preferred the data table over the other two, with one participant commenting “it is not interesting.”

Regarding trust in the recommendations, we found that trust in both the Topic Modelling classifier and the Multinomial Naïve Bayes classifier was high, at 94% and 85%, respectively.

Participants felt that these visualizations provided them with enough information to make an informed decision (Multinomial Naïve Bayes – 97%, Topic Modelling – 100%). Table 2 shows that for both types of classifiers, most of the participants preferred the stacked bar chart over the other two data representations. As might be expected, participants preferred
Table 1. Visualization Preference.

<table>
<thead>
<tr>
<th>Question</th>
<th>Stacked Bar</th>
<th>Pie Chart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do you think these visualizations increase your understanding of the recommendation?</td>
<td>77.00%</td>
<td>76.09%</td>
</tr>
<tr>
<td>How much do you trust these recommendations? (1 being not trustworthy at all to 5 being you trust this fully.)</td>
<td>3.46</td>
<td>3.41</td>
</tr>
<tr>
<td>Do you think these visualizations provide you enough information?</td>
<td>79.00%</td>
<td>77.78%</td>
</tr>
</tbody>
</table>

Table 2. Preference for specific visualizations.

<table>
<thead>
<tr>
<th>Chart Type</th>
<th>Multinomial Naïve Bayes</th>
<th>Topic Modelling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stacked Bar</td>
<td>56.10%</td>
<td>76.19%</td>
</tr>
<tr>
<td>Data Table</td>
<td>29.26%</td>
<td>7.14%</td>
</tr>
<tr>
<td>Pie Chart</td>
<td>14.63%</td>
<td>14.7%</td>
</tr>
</tbody>
</table>

the data table over the pie chart, feeling that the data table was more informative, especially for the Multinomial Naïve Bayes classifier.

A few of the participants provided answers to the question regarding if they felt that a visualization was missing something. One participant suggested that instead of showing percentage values, show the actual values as was done in the bar chart. Another participant commented that they found it to be too much work to click on the pie chart every time they wanted to see the detailed explanation for the word-based recommendations. A few participants commented that they preferred the word-based recommender to the topic-based recommender.

When examining the responses regarding trust across an individual user’s session, we observed that for the first few times that they were presented with recommendations, their level of trust was low (e.g. ratings of 2). However, as they used the plug-in more, their level of trust increased (e.g. ratings of 5) quickly.

The results from our post-usage survey showed that more than half of the participants wanted to see more than one visual representation of the recommendations. Also, the majority (75%) felt that it was “very important” or “extremely important” to represent recommendations with explanations in a visual manner.8

5. Threats to Validity

Although in our study we trained our classifier using data from a single Mozilla project - Bugzilla, we do not feel that this limits the generalizability of our results. As our focus was on the representation of the recommendations, not the accuracy of the recommendations, our results are not dependent on the project used. Similarly, the study participants were from a wide range of occupational backgrounds, which further supports the generalizability of the results. Finally, generalizability related to using an open-source project vs. a commercial project or few projects vs. many projects were not deemed to be a concern.

That the participants in the user study were not associated with the Mozilla projects may have resulted in inaccurate feedback. As this was a pilot study, we plan to address this threat in a future study where we recruit project members of the dataset used for training the recommender system. Such a study is expected to provide more detailed comments regarding trust in the recommendations and if the information provided for explanation is sufficient for the task of bug report assignment.

There is a possibility that our results may suffer from social desirability bias (i.e. “please the researcher” bias). Based on the trend where participants initially reported that they had low trust in the recommendations and then the trust level improved, we do not feel that such bias had a significant impact overall. However, we cannot discount this possibility.

6. Conclusion

This work investigated the use of visualization for explaining bug report assignment recommenders. To accomplish this, we created a web service that provides explanations of assignment recommendations for two types of recommenders using three visualizations. We found that developers did prefer visual explanations, with 75% of participants stating that the visual explanations increased their understanding of the assignment recommendations. We also found that developers gained trust in the recommendations over time and that the developers preferred a stacked bar chart.

---

8See [23] for a more detailed discussion of the study results.
References


A Practical User Feedback Classifier for Software Quality Characteristics

Rubens dos Santos¹, Karina Villela², Diego Toralles Avila¹, and Lucineia Heloisa Thom¹

¹Federal University of Rio Grande Do Sul – Institute of Informatics, Porto Alegre, Brazil,
risantos@inf.ufrgs.br, dtavila@inf.ufrgs.br, lucineia@inf.ufrgs.br
²Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany,
karina.villela@iese.fraunhofer.de

Abstract

It is common practice for users to provide feedback on apps through social media or app store reviews. This feedback is a rich source of requirements for these apps. However, manually analyzing vast amounts of user feedback is unfeasible, so automated user feedback classifiers are useful tools. This research work presents a user feedback classifier based on Machine Learning (ML) for the classification of reviews according to software quality characteristics complaint with the ISO25010 standard. We developed this approach by testing several ML algorithms, features, and class balancing techniques for classifying user feedback on a data set of 1500 reviews. The maximum F1 and F2 scores obtained were 60% and 73%, with recall as high as 94%. This approach does not replace human specialists, but reduces the effort required for requirements elicitation.

1. Introduction

Traditional Requirements Engineering (RE) techniques such as interviews and focus groups are often used to elicit the requirements of software applications. However, these techniques are not suitable for software applications whose intended users are a large, heterogeneous, geographically distributed group (the so-called crowd) [4]. On the other hand, the crowd’s opinion is accessible to software engineers in user feedback found in app stores and social media, and the RE community has acknowledged this as a relevant source of software requirements (CrowdRE) [4]. As manually analyzing vast amounts of user feedback is time-consuming and requires a lot of human effort [7], the RE community has worked on tools to automatically process user feedback and facilitate the extraction of requirements [2][6][8][9][11][15]. A particular cluster of tools consists of classifiers, i.e., tools that classify feedback into predetermined categories.

In a previous study, we performed a systematic literature review (SLR) on classifiers [13], finding a lack of studies addressing categories related to software quality. Out of 43 reviewed studies, only nine report the use of usability as a classification category. The same was found to be the case for other software quality characteristics (e.g., five studies mention performance, nine portability, and six protection). Among all studies analyzed, only three [5][11][15] use classification categories based on the ISO25010 standard [1]. One investigated all characteristics but automatically classified only usability and selected sub-characteristics [5], another reports issues in mapping spontaneous and unstructured user feedback onto the systematic structure of ISO20510 [15]. Based on this finding, we decided to work on the definition and implementation of a user feedback classifier based on Machine Learning (ML) for software quality using categories derived from ISO25010. The categories of the envisioned classifier should cover the software quality characteristics of ISO25010, but be tailored to fit the nature of user feedback. The goal is to support the elicitation of software quality requirements from a crowd by filtering non-relevant feedback and identifying feedback that might provide requirements concerning software quality characteristics.

To achieve this goal, we extracted and labeled 1500 reviews from popular apps available in the Apple App Store and Google Play. This data set was used to train and evaluate a selection of ML algorithms, features, and class balancing techniques. The very high recall (94%) of the classifier with the best evaluation results demonstrates that this classifier can successfully select feedback relevant for quality requirements.

This work is organized as follows: Section 2 presents a summary of our SLR and highlights related work. Section 3 presents our classification approach, including the definition of categories, the creation and labeling of the data set of app reviews, the training algorithm, and the evaluation of the results. Section 4 concludes this work.

DOI reference number: 10.18293/SEKE2021-055
2. Background

The SLR we reported in [13] provides a comprehensive summary of user feedback classifiers in CrowdRE, including: 1) what algorithms and features were used in each approach, 2) which kinds of user feedback were classified (e.g., Apple App Store or Google Play reviews), 3) information on the data sets used to test the classifiers, and 4) the efficacy of the results (e.g., F-measure, precision, and recall). While some approaches use dictionary-based approaches, regular expressions, or parsing, the vast majority of the reviewed work uses supervised ML. Popular ML algorithms include Naive Bayes (NB), Support Vector Machine (SVM), Logistic Regression (LR), Random Forest (RF), and Decision Trees (DT). These algorithms have been often used alongside Bag-of-Words (BOW), Stop Words, and Term Frequency–Inverse Document Frequency (TF-IDF) as ML features. When we looked at the efficacy of the results, we concluded that any of the aforementioned ML algorithms could provide good-quality results as well as poorer results. The strong variance in the setup of the studies and their efficacy suggests that it is still unclear what the most suitable ML approach for user feedback classification in a given circumstance is, and that choosing combinations of ML algorithms and ML features for the targeted circumstance still has a key role in research on such classifiers. In the following, we will focus on related work that addresses software quality.

Groen et al. [5] report on two CrowdRE studies related to software quality: 1) an exploratory study on the presence of ISO25010’s software quality characteristics in user feedback, where five people manually labeled online reviews, and 2) the identification and test of language patterns regarding usability. Similar to us, they argue that research on CrowdRE has focused on functional aspects and neglected quality aspects, but unlike the vast majority of the work on CrowdRE and from our work, they suggest using language patterns to identify quality-related statements. In any case, our work based on ML is not restricted to usability.

Lu and Liang [11] propose an ML feature called AURBoW for user feedback classification that is also based on ISO25010 quality characteristics. The authors also tailored them to better fit user feedback. The new ML feature is compared to three other features (BOW, TF-IDF and Chi Squared) in combination with three ML algorithms (NB, J48, and Bagging). Instead of proposing new classification techniques in our work, we investigated a broader set of ML algorithms, features, and class balancing techniques.

Wang et al. [15] also proposed a user feedback classifier based on ISO25010’s software quality characteristics. In their work, they tested four ML algorithms combined with TF-IDF. We tested five ML algorithms combined with three ML features and three class balancing techniques. Furthermore, Wang et al. [15] used raw ISO25010 software quality characteristics and reported problems in doing so, while we propose tailoring them in order to better address the nature of user feedback and the goals and capabilities of end users.

3. A Classifier for Software Quality

The methodology for defining and implementing our classifier consists of the following steps: First, we defined the classification categories to be used in the classifier (step 1). In parallel, we defined the criteria for the selection of user feedback to compose our data set (step 2). Afterwards (step 3), we extracted user feedback according to the criteria defined in step 2 and manually labeled the data set according to the categories defined in step 1. Then we performed a statistical analysis on the labeled data set in order to define which classification techniques to use (step 4). The next step (step 5) was to carry out an efficacy evaluation to find out the best combination of ML techniques for the data set we had created. Finally, we analyzed the results of the evaluation (step 6).

3.1. Definition of Classification Categories

Wang et al. [15] reported problems in using the ISO25010 standard to classify user feedback, such as the rare explicit reference to some ISO25010 characteristics in user feedback. In particular, they mentioned: “During the pilot labeling... we found that functional suitability, compatibility, maintainability, and security were seldom observed in app reviews.” There are several aspects to be considered to understand this phenomenon: 1) Maintainability is certainly not a concern of end users, who do not have access to the source code or sketches of software projects; 2) security has often not been one of the end users’ priorities, but their perception of its relevance is changing, especially due to the introduction of the General Data Protection Regulation in Europe; and especially 3) end users are not experts on RE or software quality and provide feedback based on their observations using their normal vocabulary. In this sense, they do not mention functional suitability or security explicitly, but rather complain about or request (security) features that, if improved or included, would increase the quality of the app in their opinion. They might not mention compatibility, but they may praise, e.g., the fact that files created using an app can be loaded into another one. The classification categories of a classifier for software quality characteristics must take into account these aspects.

As a consequence, we tailored the set of the ISO25010 standard’s software quality characteristics (functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability) to fulfill the purpose of automated classification of user feedback. Like other researchers [11, 15], we took into consideration only
the set of software quality characteristics, since user feedback often does not present enough information to allow its classification into sub-characteristics. Our tailoring consisted of: 1) excluding maintainability and 2) merging some characteristics either to increase the total number of relevant reviews per category, which contributes to the classifier’s efficacy, or to deal with the difficulty of distinguishing between them during user feedback analysis, regardless of the feedback analysis being automated or carried out by humans. Thus, compatibility and portability were merged into a single category called compatibility because both characteristics refer to the relationship between the app that is the object of the review and another element (software or hardware) in the app’s environment. Performance efficiency and reliability were also merged into performance because end users observe the behavior of apps, but usually are not capable of indicating the cause of a problem they observed [14]. For example, a frozen screen could be caused by too many users using the platform simultaneously (a performance efficiency problem) or by a software fault that was not foreseen (a reliability problem). Finally, we merged security into functional suitability, as security mechanisms perceived by end users are mostly implemented as functions that process a certain security-relevant input and provide a certain security-relevant output. The features requested in the user feedback “Why can’t I use FaceID or a password to secure the app?” are some examples.

A user review can be classified into more than one category as it may contain several statements. A review that cannot be classified into any of the four adopted categories is classified as “others”. Thus, the categories of our classifier are: functional suitability, performance, compatibility, usability, others.

### 3.2. User Feedback Selection

In this step, we determined the source of the user feedback, selected the specific apps about which to collect user feedback, and extracted a set of reviews to be manually labeled (i.e., to compose the data set). A manually labeled user feedback data set is needed to test our classifier and, as we are adopting an ML approach, also to train it.

We collected reviews about six different apps available both in the Apple App Store and Google Play (Table 1). Both platforms are prominent sources of user feedback chosen by other classifier studies [13]. We selected two popular apps from three business-related categories: Business, Productivity, and Navigation. We determined the popularity of the apps based on the Apple App Store ranking of the downloads of each of the selected app categories and the estimate provided by a list of most downloaded Google Play apps from Wikipedia [16]. We adopted this procedure to avoid sampling bias, which is a prevalent problem in collecting reviews [12]. We also deliberately avoided choosing direct competitor apps, as this work did not aim at comparing similar apps.

### 3.3. Data Extraction and Manual Labeling

We collected all reviews provided in 2017-2018 about the chosen apps together with all available metadata, which resulted in a database with 163,662 reviews. We also extracted the reviews’ star rating. Manually labeling all reviews in this database would be unfeasible. Therefore, we used simple SQL queries to randomly extract 250 reviews from each app, 50 for each star rating. The goal of selecting 50 reviews per star rating was to increase the ratio of requirements relevant reviews. For all apps chosen, most of the reviews had either 1 or 5 stars, and most of those were short and useless for requirements engineers. For example, a 5-star review that only says “Awesome” or “Cool” is not relevant for our purpose. However, training and testing data sets should include all types of reviews, so we purposefully chose not to exclude reviews with 1 or 5 stars, completely, but decreased the proportion of such reviews in our data set.

The extracted 1500 reviews were then put into a spreadsheet for the labeling process according to the categories proposed in Section 3.1. Some studies in this field split reviews into sentences before manual labeling; we did not do this because this process breaks up the context of the text.

The first author of this paper performed the labeling process alone. Therefore, we decided to perform a posterior validation of the data set labeling. In this validation, another author re-labeled a random sample of 150 reviews (10% of the data set). We analyzed inter-labeler reliability using Cohen’s kappa coefficient [10]. This coefficient was 0.59 for functional suitability, 0.65 for performance, 0.83 for compatibility, 0.84 for usability, and 0.75 for others. According to Landis and Koch [10], the interpretation of these values is as follows: moderate agreement for functional suitability, substantial agreement for performance and the category “others”, and almost perfect agreement for compatibility and usability. These results suggest that our data set is consistent even though inter-labeler agreement varies among categories.

### 3.4. Statistical Analysis

After labeling the data set, we performed a statistical analysis to understand its characteristics and facilitate the next
steps. Figure 1 shows the distribution of the reviews among the classification categories. One should keep in mind that a review may contain several statements and therefore be classified into several categories. According to Fernandez et al. [3], balanced data sets are preferred for training ML algorithms, i.e., in binary classifications such as ours, half of the reviews should be classified within a category and half outside of it. As seen in Figure 1, our data set is imbalanced, but an imbalanced training data set can be treated with class balancing techniques [3].

Figure 2 shows for each app the average star rating of the reviews classified into each category. This kind of analysis, performed here on the manually labeled data set but to be supported by our classifier later, allows identifying an app’s strengths and weaknesses. Users are, e.g., satisfied with the performance of Google Drive and the usability of Indeed Job Search. Categories with a low average star rating indicate opportunities for improvement, which can then be investigated in-depth. This kind of information is very valuable to support the evolution of the analyzed apps or the development of competing apps. As 1500 random reviews may not be enough to perform such an analysis, Figure 2 is only illustrative.

Furthermore, we analyzed the words most correlated to each category using the Chi Squared technique. The results are shown in the word clouds in Figure 3 and provide additional validation of the manual labeling. As expected, Figure 3a shows that the words “feature”, “ability”, “able”, and “option” are very correlated to functional suitability. The word “password” is also among the correlated words, which makes sense as we merged functional suitability with security. Moreover, the other correlated words refer to specific app features, for example “search”, “notifications”, “upload”, and “email”. Figure 3b shows words correlated to usability such as “hard”, “friendly”, “intuitive”, “confusing”, “easy”, and “interface”, whereas Figure 3c shows that the words “battery”, “network”, “crashing”, “slow”, “sync”, and “time” are closely correlated to performance. Finally, Figure 3d shows that when users want to talk about portability or compatibility, they usually mention devices, platforms or other apps that they want to use or are using together with the app that is the object of the review.

3.5. Automated Classification and Evaluation

This work aimed to discover the best combination of ML algorithms, features, and class balancing techniques for automatically classifying user feedback into software quality characteristics. As it would not be feasible to test all possible combinations within the time and effort constraints, we analyzed NB, LR, DT, RF, and SVM as ML algorithms and BOW, TF-IDF, and Stop Words as ML features because they yielded the most relevant evaluation results in our SLR [13] and were available in the SciKit library. Furthermore, we searched the literature for methods to solve the class imbalance problem, finding the following class balancing techniques: undersampling, SMOTE, and Cost-Sensitive Learning (CSL) - Balanced, 1:2, 1:5 and 1:10 [3]. Such class balancing techniques were only applied in the training data set.

The automated classification consisted of exhaustively testing all combinations of the selected techniques and generating their evaluation metrics. As Stop Words is a secondary feature, it was used in all combinations.

<table>
<thead>
<tr>
<th>Algorithm 1: Automated Classification and Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>initialize final confusion matrices;</td>
</tr>
<tr>
<td>for every combination of classification techniques do</td>
</tr>
<tr>
<td>initialize intermediate confusion matrices;</td>
</tr>
<tr>
<td>for 10 times do</td>
</tr>
<tr>
<td>shuffle the data set;</td>
</tr>
<tr>
<td>initialize partial confusion matrices;</td>
</tr>
<tr>
<td>for every fold from 10-fold cross-validation do</td>
</tr>
<tr>
<td>train classifier with the other 9 folds;</td>
</tr>
<tr>
<td>generate predictions for test fold;</td>
</tr>
<tr>
<td>compute partial confusion matrix from predictions;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>add partial confusion matrices into a intermediate confusion matrix;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>add intermediate confusion matrices into a final confusion matrix;</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>calculate evaluation metrics from final confusion matrices;</td>
</tr>
</tbody>
</table>

We used 10-fold cross-validation in this study. The complete pseudo-code is shown in Algorithm 1. The code was im-

https://scikit-learn.org/stable/
3.6. Analysis of Results

Tables 2 and 3 show the best results obtained from the application of the ML techniques to our data set. Table 2 shows the best F1 scores for each class balancing technique and each category, whereas Table 3 shows the best F2 scores. The corresponding ML model is included in parenthesis following the format: (algorithm/feature). The maximum scores per category are highlighted in bold. It is interesting to observe how much the class balancing techniques increased the classification efficacy. Table 2 shows relatively low maximum F1 for each category. Comparing our maximum F1 scores to the maximum F1 scores obtained by classifiers used in similar studies (65.4% in [11] and 62.8% in [15]), we found that our F1 results were not satisfactory. Hence, it is not possible to use the ML model highlighted in bold in Table 2 to replace human specialists, which would not be possible anyway due to the complexity of the requirements elicitation task. We propose using F2 to select the best ML models because F2 emphasizes the ML model’s recall more than its precision. Ensuring a low number of false negatives is more important when supporting human specialists in deep investigation of potential requirements. Our F2 scores are satisfactory, showing great recall measures. For example, the best classifier for functional suitability had an F2 score of 0.73 with 94% recall. With the support of our classifier, almost no relevant reviews will be lost due to classification mistakes (false negatives). Hence, we conclude that our classifier is capable of helping specialists focus on reviews that can provide quality requirements without causing loss of information.

4 Conclusion

In this work, we presented a user feedback classifier for software quality characteristics based on the ISO25010 standard [1]. In order to implement it using ML techniques, we manually classified 1500 reviews. The data set was highly imbalanced, which represents a true challenge in the field of ML classification. To address this problem, we adopted three class balancing techniques in our investigation: undersampling, SMOTE, and CSL.

We also performed statistical analyses on this data set, showing, e.g., the words most closely correlated with each category, which confirmed the quality of our manual labeling of the reviews and gave an idea of the kind of analyses the results of an automated user feedback classifier can support.

Our approach consisted of investigating the efficacy of different combinations of ML algorithms, features, and techniques found in the literature. The final results of the
Table 2: Best F1 score for each class balancing technique according to the classification categories.

<table>
<thead>
<tr>
<th>None</th>
<th>Functional Suit.</th>
<th>Performance</th>
<th>Compatibility</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.52 (SVM/TFIDF)</td>
<td>0.52 (SVM/TFIDF)</td>
<td>0.52 (SVM/TFIDF)</td>
<td>0.52 (SVM/TFIDF)</td>
<td>0.52 (SVM/TFIDF)</td>
</tr>
<tr>
<td>Undersampling</td>
<td>0.55 (SVM/BOW)</td>
<td>0.55 (SVM/BOW)</td>
<td>0.54 (SVM/BOW)</td>
<td>0.52 (SVM/BOW)</td>
</tr>
<tr>
<td>SMOTE</td>
<td>0.50 (SVM/BOW)</td>
<td>0.50 (SVM/TFIDF)</td>
<td>0.50 (SVM/TFIDF)</td>
<td>0.50 (SVM/TFIDF)</td>
</tr>
<tr>
<td>CSL (Balanced)</td>
<td><strong>0.60 (LR/TFIDF)</strong></td>
<td>0.54 (SVM/TFIDF)</td>
<td><strong>0.59 (DT/TFIDF)</strong></td>
<td>0.52 (LR/TFIDF)</td>
</tr>
<tr>
<td>CSL (1:2)</td>
<td>0.55 (LR/TFIDF)</td>
<td>0.53 (SVM/TFIDF)</td>
<td>0.53 (SVM/TFIDF)</td>
<td>0.52 (SVM/BOW)</td>
</tr>
<tr>
<td>CSL (1:5)</td>
<td>0.59 (LR/TFIDF)</td>
<td><strong>0.57 (LR/TFIDF)</strong></td>
<td>0.56 (LR/TFIDF)</td>
<td><strong>0.55 (LR/TFIDF)</strong></td>
</tr>
<tr>
<td>CSL (1:10)</td>
<td>0.57 (LR/BOW)</td>
<td>0.56 (LR/BOW)</td>
<td>0.56 (LR/BOW)</td>
<td>0.55 (LR/BOW)</td>
</tr>
</tbody>
</table>

Table 3: Best F2 score for each class balancing technique according to the classification categories.

<table>
<thead>
<tr>
<th>None</th>
<th>Functional Suit.</th>
<th>Performance</th>
<th>Compatibility</th>
<th>Usability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.50 (SVM/BOW)</td>
<td>0.49 (SVM/BOW)</td>
<td>0.49 (SVM/BOW)</td>
<td>0.48 (SVM/BOW)</td>
<td>0.48 (SVM/BOW)</td>
</tr>
<tr>
<td>Undersampling</td>
<td>0.53 (SVM/BOW)</td>
<td>0.55 (SVM/BOW)</td>
<td>0.55 (SVM/BOW)</td>
<td>0.54 (SVM/BOW)</td>
</tr>
<tr>
<td>SMOTE</td>
<td>0.54 (SVM/BOW)</td>
<td>0.53 (SVM/TFIDF)</td>
<td>0.53 (SVM/TFIDF)</td>
<td>0.52 (SVM/TFIDF)</td>
</tr>
<tr>
<td>CSL (Balanced)</td>
<td>0.65 (LR/TFIDF)</td>
<td>0.61 (LR/TFIDF)</td>
<td>0.61 (LR/TFIDF)</td>
<td>0.59 (LR/TFIDF)</td>
</tr>
<tr>
<td>CSL (1:2)</td>
<td>0.54 (LR/TFIDF)</td>
<td>0.51 (SVM/TFIDF)</td>
<td>0.50 (SVM/BOW)</td>
<td>0.50 (SVM/BOW)</td>
</tr>
<tr>
<td>CSL (1:5)</td>
<td>0.72 (LR/TFIDF)</td>
<td>0.66 (LR/TFIDF)</td>
<td>0.63 (LR/TFIDF)</td>
<td>0.60 (LR/TFIDF)</td>
</tr>
<tr>
<td>CSL (1:10)</td>
<td><strong>0.73 (LR/TFIDF)</strong></td>
<td><strong>0.70 (LR/TFIDF)</strong></td>
<td><strong>0.68 (LR/TFIDF)</strong></td>
<td><strong>0.65 (LR/TFIDF)</strong></td>
</tr>
</tbody>
</table>

automated classification are almost equivalent to those of other studies (e.g., F1 score of 60% for functional suitability against 62.8% in [15] and 65.4% in [11]).

There is still room for improvement in our work. The data set could be labeled manually by a second specialist and possible inconsistencies could be discussed, which could make the manual classification even more reliable. Furthermore, new training techniques have been proposed during the development of this work, meaning that we could extend our automated classification and evaluation to include them. Finally, this work showed that our classifier cannot replace human specialists, but it can significantly reduce the number of reviews that need to be analyzed manually without causing loss of information, which means that less effort is required from specialists.

Acknowledgments

This study was part of a bachelor’s thesis and was partially funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. Diego Toralles Avila is a CAPES scholarship holder through the Programa de Apoio à Pós-Graduação (PROAP).

References


Improved Multiple Part Algorithm (IMPA) to extract multiple solutions for RNA sequence classification problem

Naoual Guannoni*, Faouzi Mhamdi†, Mourad Elloumi‡

*†‡ Laboratory of Technologies of Information and Communication and Electrical Engineering (LaTICE), National Higher School of Engineers of Tunis (ENSIT), University of Tunis, Tunisia
*Faculty of Science of Tunis, University of Tunis el Manar, Tunis, Tunisia
† Higher Institute of Applied Languages and Computer Science of Beja, University of Jendouba, Tunisia
‡ naouel.gannouni90@gmail.com, faouzi.mhamdi@ensi.rnu.tn, Mourad.Elloumi@gmail.com

Abstract—The methods, which extract knowledge from Next Generation Sequencing Data (NGS) are highly requested nowadays. The attention to analysis biomedical data is increasing proportionally. In this work, we focus to elicit and discover a higher amount of knowledge by computing many classification models in a single run, and therefore to identify most of the features related to an investigated class. Major efforts have been made in this field and a last algorithm is proposed “Multiple Part” for data analysis and extraction of new and more knowledge from them. In this paper, we propose a new version of Multiple Part algorithm which integrates a heuristic evaluation method and a feature elimination technique in order to extract multiple and equivalent solution for biomedical data. In order to prove the validity of our algorithm, we analyze an RNA-seq of cancer diseases data sets extracted from The Cancer Genome Atlas (TCGA). Furthermore, we validate our approach by comparing it with the existing methods. Experimental results show the efficacy of our proposed algorithm.

Index Terms—Multiple solution, Multiple Part, Camur, Merit, heuristic method, IMPA.

I. INTRODUCTION

The cancer mechanism becomes more worldwide major public health issue. Since cancer is one of the leading causes of mortality, many researches have been developed in order to understand its mechanisms and discover new knowledge to prevent and to treat this serious disease [1]. In recent years RNA-seq protocol counting the RNA fragments that are aligned on a reference genome. In this scenario, it is important to identify informative genes with high prognostic value to distinguish between healthy tissue and tumoral tissue types. In this work, we focus on the amelioration and the adoption of a new algorithm for classifying RNA-seq case-control samples, which is able to compute multiple human readable classification models. In the past, such problems have been solved by the use of supervised and unsupervised machine learning algorithms such as decision tree, rule-based, ensembles decision tree, neural networks and Support Vector Machines (SVM) [2] [3] [4] [5]. These techniques have been also used to improve diagnosis of diseases such as Alzheimer, Breast Cancer or Meningitis [6] [7] [1]. The big limits with the application of these machine learning algorithms are related to the managing of the huge amount of data. In fact, for biological datasets, a high learning time is needed for data analysis and the extraction of new knowledge from them [8]. Also, all these classical algorithms compute just a single classification method that contains few of features. While our goal is the extraction and the discovery of the maximum knowledge from these RNA sequence datasets by computing many alternative and equivalent classification models.

Multiple and equivalent solutions extraction from biological datasets is a novel concept which has recently caught the attention of researchers. Obtaining a set of efficient solutions with a better compromise between the features and with a reduced running time is the goal of this study. More details about these works are presented in the rest of the paper. All these methods can on one side provide a relevant number of rules (solutions) with low performance. On the other side, the number of extracted rules at each iteration can be insufficient compared to the big RNA-sequence datasets used.

In this work, we propose a new algorithm to optimize Multiple Part algorithm for classifying RNA-seq case-control samples. This algorithm integrates a discretization method, a feature elimination technique and a heuristic evaluation method for each subset of selected features. The final aim of this work is to provide a more compact, human interpretable models that can aid biologists or doctors to make a decision about the classification of diseases. The rest of this article is organized as follows. In section 2 we present a literature review about methods to extract multiples and equivalent solutions. In section 3 we describe our proposed algorithm (IMPA). The experiments and their results are discussed in section 4. Finally, in section 5, we report the conclusion and we present future works.

II. LITERATURE REVIEW

A number of previous studies have been focused on the extraction of multiple and equivalent solutions in biomedical data classification problems. One approach is presented in Fiscon et al. [9], where the authors proposed meta-heuristic approach based on an evolutionary algorithm to find a solution for identifying a large number of small species-specific genomic
subsequences. One other work proposed by Gholami et al., [10], this classification-based approach is based on recursive feature elimination RFE method. The limit of this algorithm is that at each iteration, only a single variable should be chosen to remove. This would be inefficient in many high dimensional applications such RNA-sequence datasets.

In recent years, several works pointed to extract multiple solutions interpretable by human using rule and tree-based classification algorithms. Valerio et al., [11], proposed a new algorithm Camur (Classifier with a Alternative and Multiple Rule-based model). This algorithm able to extract, multiple, alternative and equivalent rule-based models (Ripper). These rules represent the most relevant set of features related to the case and control samples. In 2016, Fiscon et al., [9], proposed a metaheuristic approach in order to find solutions for identifying a large number of small species-specific genomic subsequences. This approach aims to extract multiple solutions using rule and tree-based classification algorithms. In 2017, Fabrizio Celli, et al., [1] developed a new algorithm called Big Biomedical data classifier (BIGBIOCL). This algorithm able to classify a large DNA methylation dataset. BIGBIOCL is inspired by Camur algorithm in order to apply classification methods to big datasets. In 2019, Guannoni et al., [8] proposed a new method that extracts multiple and equivalent classification methods. This method Called “Multiple Part” algorithm that integrates rule-based classification method (Part) and a feature elimination technique in order to obtain more interpretable models in a reduced execution time. In the first, this method iteratively computes the rule-based classifier, then it computes the power set of the features present in the rules, iteratively eliminates these combinations from the data sets, and execute again the classification procedure until a stopping criterion is verified. Experimental results show that “Multiple part” is an important algorithm for extracting multiple, equivalent and alternative solution in a reduced execution time.

III. THE PROPOSED ALGORITHM: IMPROVED MULTIPLE PART ALGORITHM (IMPA)

We propose an enhanced version of “Multiple Part algorithm” which specifies the quality of each combination of the features found in the rule using an heuristic evaluation method. We called our proposed method as IMPA (Improved Multiple Part Algorithm). IMPA is new algorithm inspired by “Multiple Part” in order to extract multiple and equivalent solutions with higher performance and in few time executions. It is a tool to obtain knowledge by extracting several alternative classification models for gene features in RNA-seq data. Through evaluation of the possible combination to delete, and through iterative deletion of selected features, extraction of equivalent classification models is possible using IMPA algorithm. The implementation of our new algorithm is essentially based on feature elimination method by evaluating each power set of features. One of the reasons is that the merit function for evaluation the set of features enables to evaluate the worth of a subset of features by considering the individual predictive ability of each feature as well as with the degree of redundancy between them.

A. Steps of the IMPA algorithm

IMPA implements the following steps:

1) Compute Rule-based method (PART): our algorithm executes at first a rule-based algorithm (Part) that extract a set of logic rules “if CONDITION then CLASS” rules which provide an immediate relationship between the class and one or more features (genes).

2) Computes the power set of the features present in the rule: IMPA calculates the power set of the features present in the rule after each iteration. Then, all the combination are stored in a memory list.

3) Discretize the data set: to computes the score of each combination, we need to discretize continuous features. A copy of the training data is first discretized then passed to compute the quality of each combination features. In this work we choose to use the discretization method of Fayyad and Irani [12] because it has been showed that the number of classification errors generated by this method is comparatively smaller than the number of errors generated by the other discretization algorithms.

4) Compute the quality of each combination features using merit function: we use a correlation based heuristic evaluation function for computing the score of each set of combination feature. This function called “Merit function”. Merit function is a measure that calculates feature-class and feature-feature correlations using a measure called symmetrical uncertainty (SU) correlation. This function enables to evaluate the heuristic “merit” of feature subsets. It ranks the feature subsets according to a correlation based heuristic evaluation function [13]. The subset with the lowest merit is considered the first combination to be eliminated from the data set at time. let Pk is a subset of features (one combination), we define the Merit function associated with Pk as follows:

\[
\text{Merit}(P^k) = \frac{j * r_{yx}}{\sqrt{j + j * (j - 1) * r_{xx}}}
\]  

(1)

where j=|Pk| is the number of subset features, \( r_{yx} \) is the average of the correlations between the subset features and the class and \( r_{xx} \) is the average inter-correlation between subset features.

The numerator of Equation 1 can be considered to provide an indication of the predictive of the class a set of features are; The denominator represents how much redundancy there is among the features [13]. Merit function uses SU to measure correlation. SU [14] associated with two features \( x_1 \) and \( x_2 \) is defined by:

\[
SU(x_1, x_2) = 2 * \left[ \frac{GI}{H(x_1) + H(x_2)} \right]
\]  

(2)

more details about the SU function is presented in [14]. The advantage of the Merit function is that is allows to
compare subsets of feature in different sizes. Thus, it allows to evaluate the contribution of a new feature.

5) Scores all possible features combination: after computing a score of each possible combination, we sort the list of combination in ascending order according to the score (The worst Merit to the best Merit).

6) Perform feature elimination method: eliminates all the possible combinations of features by starting with the worst Merit and run the analysis again at each time. The feature elimination is iterated in two execution-mode:

• A loose feature elimination mode: in the first, a classification with the PART algorithm is performed. This mode takes the results from the first classification and build the combinations (power set) of the found features, whose combinations are iteratively eliminated according to the worst score from the data set. After each elimination of the feature combination, a classification step is built. The new extracted features that are present in the current classification model are added to the features list and are going to be processed in the next iterations. In loose mode, once a feature is removed it inserted again in the data set.

• A strict feature elimination mode: in the first, a classification with the PART algorithm is performed. The features appear from the first model are extracted and then eliminated one by one according to the worst score. A classification is iterated after each elimination on the resulting data set. In the strict mode, once a feature is eliminated it is never inserted again in the data set.

7) Our proposed algorithm performs again the classification procedure until a stopping criterion is verified: the reliability (F-measure) < a given threshold, maximum number of iterations (Max-iter) is reached, or the list of features has been completely treated.

In the final, we obtain a several of relevant number of equivalent classification models e.g., "IF feature <1.50 then the sample is NORMAL" with higher performance. These rules composed of a list of relevant genes related to a particular class.

B. Execution example of our IMPA algorithm

Given a data set of RNA-seq data related to Breast cancer with two class tumoral and normal:

• IMPA extracts through the first execution a model composed of a set of rules, e.g., (\(ADHFE1 \geq 4.69\)) AND (\(ACSBG1 \geq 0.37\)) OR (\(HBBP1 \leq 0.04\)) then normal.

• The rules contain a set of three features (genes) \(S1 = \{ADHFE1, ACSBG1, HBBP1\}\).

• The power set is computed: \(P1 = \{\{ADHFE1\}, \{ACSBG1\}, \{HBBP1\}, \{ADHFE1, ACSBG1\}, \{ADHFE1, HBBP1\}, \{ACSBG1, HBBP1\}, \{ADHFE1, ACSBG1, HBBP1\}\} \).

• Discretize the dataset.

• Compute the quality of each combination features using merit function. \(P1 = \text{Merit} \{\{ADHFE1\}\} = 0.7, P2 = \text{Merit} \{\{ACSBG1\}\} = 0.3, P3 = \text{Merit} \{\{HBBP1\}\} = 0.1, P4 = \text{Merit} \{\{ADHFE1, ACSBG1\}\} = 0.04, P5 = \text{Merit} \{\{ADHFE1, HBBP1\}\} = 1.18, P6 = \text{Merit} \{\{ACSBG1, HBBP1\}\} = 0.130, P7 = \text{Merit} \{\{ADHFE1, ACSBG1, HBBP1\}\} = 0.175\)

• Sort the list of combination in ascending order: \(\{P4, P3, P6, P7, P5, P2, P1\}\).

• The first item of the power set is eliminated from the data set and the classification is performed, which provides a new set of features, \(S2 = \{HBBP1, ADH4\}\).

• The first power set \(P1\) is completely performed.

• After the treating of \(P1\), the power set \(P2\) from \(S2\) is computed and the classification is performed.

• The algorithm continues again the classification procedure until one of the stopping criteria is verified.

IV. EXPERIMENTAL STUDY AND RESULTS

A. Description of the dataset

Our experimental analysis is focused on RNA-seq data related to Breast cancer disease (BRCA). These data are extracted from public available data of The Cancer Genome Atlas (TCGA) [15] [16]. The data set of BRCA composed of a matrix in comma separated value format, which is the input of our algorithm. The rows of the matrix correspond to a set 59 samples that represent the sequenced tissues of the patients. The columns correspond to 20532 features which represent the gene expression profile. The last column represents the class e.g., normal - tumoral. Each cell contains the gene expression measure Reads Per Kilobase per Million mapped reads (RPKM) value for each gene expression measure. [17].

<table>
<thead>
<tr>
<th>Sample ID</th>
<th>ANO8</th>
<th>Clorf27</th>
<th>TRPM6</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>A8-A09D</td>
<td>2.64</td>
<td>5.42</td>
<td>0.38</td>
<td>Breast cancer</td>
</tr>
<tr>
<td>BHI-A0DH</td>
<td>1.46</td>
<td>6.47</td>
<td>0.76</td>
<td>Normal</td>
</tr>
<tr>
<td>GM-A2DB</td>
<td>3.13</td>
<td>14.21</td>
<td>0.61</td>
<td>Breast cancer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GM-A2DB</td>
<td>3.86</td>
<td>5.15</td>
<td>0.59</td>
<td>Breast cancer</td>
</tr>
</tbody>
</table>

Concept and experimental study: We compare in the experimental study the obtained results of IMPA with results of CAMUR and "Multiple Part" [8]. Our comparison will be based on the number of extracted models, the performance of the models, the number of relevant features and the execution time. In fact, our goal is to validate a new supervised classification algorithm able to extract multiple models by building hundreds of classification iterations on a massive number of relevant features in few hours. We choose to variate the iteration numbers (Iter-nb) between 20 and 150. Also, we variate the minimum number of F-measure on 0.8 and 0.9. We use for each parameter the two-execution mode” strict” and“ loose”. For evaluating the classification models, we adopt the accuracy and the F -measure equations. –Our proposed algorithm is implemented in JAVA language programming. The experimentation has been executed on a laptop with an on 2.71 GHz Intel (R) Core (TM)i7 CPU and 32 GB of RAM. Table III reports the genes that are most represented in the models. Table IV, Table V and Table VI represent the classification result, the number of extracted rule (Nb-rule), the number of relevant features in few hours. We choose to variate the iteration numbers (Iter-nb) between 20 and 150. Also, we variate the minimum number of F-measure on 0.8 and 0.9. We use for each parameter the two-execution mode” strict” and" loose". For evaluating the classification models, we adopt the accuracy and the F -measure equations. –Our proposed algorithm is implemented in JAVA language programming. The experimentation has been executed on a laptop with an on 2.71 GHz Intel (R) Core (TM)i7 CPU and 32 GB of RAM. Table III reports the genes that are most represented in the models. Table IV, Table V and Table VI represent the classification result, the number of extracted rule (Nb-rule), the number
TABLE II: Classification rules example extracted from Camur, multiple part and IMPA with a classification accuracy \( \geq 90\% \).

<table>
<thead>
<tr>
<th>Extracted rules of Camur</th>
<th>Extracted rules of Multiple Part</th>
<th>Extracted rules of IMPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ADH4</td>
<td>127 ≥ 0.26) AND (ADHC1</td>
<td>27245 ≥ 11.02) ⇒ normal</td>
</tr>
</tbody>
</table>
| Extracted rules of IMPA algorithm: extracted genes related to breast cancer.

<table>
<thead>
<tr>
<th>Features</th>
<th>Occurence</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAG1AP1 55974</td>
<td>5</td>
</tr>
<tr>
<td>HOXA7 3204</td>
<td>5</td>
</tr>
<tr>
<td>CDC A855143</td>
<td>5</td>
</tr>
<tr>
<td>LOC 572558</td>
<td>3</td>
</tr>
<tr>
<td>RERGL 79785</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 1: The process of IMPA algorithm.

V. DISCUSSION

From the tables we can conclude some considerations regarding the link between the number of extracted rules, the number of features, the accuracy of the rules and the execution time. The running time of our IMPA algorithm is faster than Camur but not for the "multiple Part". Multiple Part remain the faster because it uses a faster method for extracted rules of extracted features (Nb-f), the average accuracy (Aver-acc) of each rule and the running time of each classification.

On the other hand, as shown in Figure 3, The accuracy of extracted rules of IMPA in all cases is in the range [0.99, 1]. This mean that it extract always compact rules with the higher performance compared to the other algorithms. As shown in Table III, the most represented genes extracted from the rules are RAG1AP1 with id 55974, HOXA7 with id 3204 and CDCA with id A855143. These genes are the most involved in the breast cancer classification models. Many studies have shown that HOXA7 plays a critical role in regulating the proliferation of estrogen receptor -positive cancer cells [18]. A recent study shows that RAG1AP1 is the new biomarker candidate of breast cancer development [19]. Another study shows that CDCA plays a crucial role for the prevention of this disease [20]. We can conclude that such information in the extracted rules IMPA can be considered as an important result to help biologists and doctors in analyzing the genetics of breast cancer disease.

Using loose feature elimination mode (Table V), all the algorithms completed all the iterations but they extract only a few numbers of extracted rules. The cause can be that after the first execution, the extracted rules do not exceed the f-measure value (0.8). In Table V, since almost all the algorithms provide
Fig. 2: Execution time of Camur, Multiple Part and IMPA.

Fig. 3: Average Accuracy of extracted rules of Camur, Multiple Part and IMPA

Fig. 4: Extracted rule number of Camur, Multiple Part, IMPA

a few numbers of rules, our IMPA algorithm provides more rule number compared to the other algorithm and there are not classification errors compared to CAMUR. In Table VI the number of iterations is not treated for CAMUR because the stopping criteria is reached (f-measure smaller than 0.9). For multiple Part and IMPA, all the iteration are executed but they give all a few numbers of rules. The cause can be that the extracted rules does not contain the features and therefore the power set list to be removed is empty for each iteration. Therby, IMPA algorithm provides more several efficient classification rules with high performance and remains faster then the other algorithms.

From this detailed analysis we can conclude that our IMPA algorithm is an elegant algorithm that is able to extract more multiple classification models with high accuracy for the RNA-sequence classification problem. Therefore, it enables to identify most of the features related to the investigated class. Our algorithm operating efficiently because the integration of the heuristic method to evaluate the feature subset which is based it can help to provide more accuracy human interpretable models. Moreover, our algorithm can be efficient and can provide thousands of equivalent solutions in one single run.

VI. CONCLUSION

In this work, we presented a new algorithm (IMPA) enables to extract multiple and equivalent models for RNA-sequence classification problem. Our proposed algorithm adopted a feature elimination technique and integrated an heuristic evaluation method for each subset of selected features in order to provide more accuracy rules for each classification model. IMPA is applied on a set of RNA-seq data focusing on Breast cancer from TCGA. After the experimental study, we prove that our proposed algorithm is a reliable technique for extract more compact rules with more relevant features than multiple Part and CAMUR. It can also ignore redundant and duplicate rules to be executed when ordered and evaluates the power set of features to be eliminated.

In a future work, we plan to more ameliorate the execution time of our algorithm to be applied on big data set. As another future work we can extend the analyses to another biological data set, e.g., RNA-sequce data of COVID-19, DNA-methylation values and DNA-Barcoding in order to confirm the validity of our approach. Also, we are investigating the possibility to validate the extracted genes by domain experts with deep analysis.

REFERENCES


<table>
<thead>
<tr>
<th>Parameters</th>
<th>Camur</th>
<th>Multiple Part</th>
<th>proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter-Nb=20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>18</td>
<td>0.96</td>
<td>22</td>
</tr>
<tr>
<td>Iter-Nb=40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>0.96</td>
<td>47</td>
</tr>
<tr>
<td>Iter-Nb=60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>0.97</td>
<td>82</td>
</tr>
<tr>
<td>Iter-Nb=80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>80</td>
<td>0.96</td>
<td>96</td>
</tr>
<tr>
<td>Iter-Nb=100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>90</td>
<td>0.97</td>
<td>103</td>
</tr>
<tr>
<td>Iter-Nb=120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>110</td>
<td>0.99</td>
<td>123</td>
</tr>
<tr>
<td>Iter-Nb=140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>130</td>
<td>0.99</td>
<td>143</td>
</tr>
<tr>
<td>Iter-Nb=150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>140</td>
<td>0.99</td>
<td>153</td>
</tr>
</tbody>
</table>

TABLE IV: Results of classification analysis with Camur, Multiple Part and IMPA using Strict execution mode (F-measure=0.8).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Camur</th>
<th>Multiple Part</th>
<th>proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter-Nb=20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>0.972</td>
<td>3</td>
</tr>
<tr>
<td>Iter-Nb=40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Iter-Nb=60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>2</td>
<td>0.958</td>
<td>2</td>
</tr>
<tr>
<td>Iter-Nb=80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>2</td>
<td>0.958</td>
<td>2</td>
</tr>
<tr>
<td>Iter-Nb=100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0.972</td>
<td>5</td>
</tr>
<tr>
<td>Iter-Nb=120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>4</td>
<td>0.958</td>
<td>4</td>
</tr>
<tr>
<td>Iter-Nb=140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>4</td>
<td>0.958</td>
<td>4</td>
</tr>
<tr>
<td>Iter-Nb=150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>4</td>
<td>0.972</td>
<td>3</td>
</tr>
</tbody>
</table>

TABLE V: Results of classification analysis with Camur, Multiple Part and IMPA using loose execution mode (F-measure=0.8).

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Camur</th>
<th>Multiple Part</th>
<th>proposed method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iter-Nb=20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>0.97</td>
<td>22</td>
</tr>
<tr>
<td>Iter-Nb=40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>24</td>
<td>0.982</td>
<td>32</td>
</tr>
<tr>
<td>Iter-Nb=60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>20</td>
<td>0.98</td>
<td>21</td>
</tr>
<tr>
<td>Iter-Nb=80</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>46</td>
<td>0.97</td>
<td>60</td>
</tr>
<tr>
<td>Iter-Nb=100</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>52</td>
<td>0.98</td>
<td>70</td>
</tr>
<tr>
<td>Iter-Nb=120</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>63</td>
<td>0.97</td>
<td>92</td>
</tr>
<tr>
<td>Iter-Nb=140</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>140</td>
<td>49</td>
<td>0.96</td>
<td>35</td>
</tr>
<tr>
<td>Iter-Nb=150</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>150</td>
<td>67</td>
<td>0.968</td>
<td>78</td>
</tr>
</tbody>
</table>

TABLE VI: Results of classification analysis with Camur, Multiple Part and IMPA using Strict execution mode (F-measure=0.9).


Attention Guided Filter for Jointly Extracting Entities and Classifying Relations

Shaoze Chen¹, Su Wang², Wenxin Hu*²
¹School of Computer Science and Technology, East China Normal University, Shanghai, China
²School of Data Science Engineering, East China Normal University, Shanghai, China

Abstract—Jointly extracting entities and classifying relations aims to detect all possible triples from unstructured text with a single model. Tagging-based method effectively improves the performance of jointly relation extraction. However, some tagging-based approaches ignored that one entity pair may exist multiple relations and others set an empirical threshold value for selecting one or more relevant relations, which becomes the bottlenecks of the model. As a solution, we propose the attention guided filter, namely, AGFRel, which introduces transformer blocks to learn the number of relations for every entity pair to filter out irrelevant relations. Moreover, each module of the model has a multi-head attention guided layer to highlight valuable information. Extensive experimental results show that AGFRel is capable of gaining better performance on various tasks including overlapping triples extraction and multiple triples extraction. On NYT and WebNLG public datasets, our model obtains F1 score 90.8 and 91.9 respectively and achieves a new state-of-the-art performance.

Index Terms—transformer, attention mechanism, joint extraction model, NYT, WebNLG

I. INTRODUCTION

Relation Extraction aims to extract relational triples from unstructured text. It plays a crucial role in many applications of natural language processing, such as biomedical knowledge discovery [1] and knowledge base construction [2].

Pipeline-based approach is an intuitive method to extract triples. It first recognizes all entity mentions in the text and then classifies relations for each entity pair. This method mechanically decomposes the relation extraction task into two independent sub-tasks: named entity recognition (NER) and relation classification (RC), which ignores the relevance between them and results in error propagation [3]. The joint extraction models are proposed to tackle this problem. It can share information and simultaneously extract entities and relations in a single model. The first joint tagging-based model is proposed by [4], which sets a new label for each token containing entity position information and relation type information. This tagging-based method achieves a significant improvement but cannot solve the overlapping problem: two relational triples share one or two entities. [5] further divides the overlapping problem into three scenarios (see Figure 1): Normal, SingleEntityOverlap (SEO) and EntityPairOverlap (EPO). [6, 7] propose different decomposition strategies and tagging schemes to handle SEO cases. Although these models can achieve better performance, they ignore the EPO cases that an entity pair has multiple relations. [8] makes an attempt to handle EPO cases, which artificially sets a threshold to determine the number of relations in entity pairs. Such an approach results in a situation where manual adjustment requires extra workload to achieve good performance on a particular dataset. Meanwhile, a fixed threshold affects the performance of relation classification and generalization of the model. On the other hand, these models cannot handle the nested entity problem since their tagging scheme merely serves as a yes or no decision when they detect the span of entities. The problem of missing nested entities misleads the model to extract wrong triples.

In this paper, we propose a novel method to extract overlapping triples and handle the nested entity problem. Our main idea is to predict the number of entities and relations firstly, and then derive the corresponding triples. More precisely, our model is composed of two modules: subject extractor (SE) and relation extractor (RE). Each module of our model has a multi-head guided layer to filter out useless information and highlight valuable information. SE aims to extract all possible subjects. RE is comprised of two components. The transformer component is used to recognize the number of relations under a specific subject, and the other component predicts the probability distribution of relations. We propose a novel tagging scheme that annotates the relation numbers at the start and end positions. In the tagging process, if the number of relations is greater than 0, the token is regarded as an object candidate. We then use the nearest strategy to detect the span of entities. Finally, one or more relations in entity pairs are derived from mapping the probability distribution of relations. We evaluate our model on NYT [9] and WebNLG.

![Fig. 1. Examples of Normal, SingleEntityOverlap (SEO) and EntityPairOverlap (EPO) classes.](image)

*Corresponding author: wxhu@cc.ecnu.edu.cn
The source code of this paper: https://github.com/almbreeder/AGFRel
DOI reference number: 10.18293/SEKE2021-153
[10] public datasets and experiments show that our model has better performance than previous models.

In general, our contributions are as follows:

- We propose a novel model, Attention Guided Filter (AGFRe), which can learn the exact number of relations in the sentences. It means that the model can solve the overlapping problem in different scenarios.
- We adopt a novel tagging scheme to handle the nested entity problem. Our tagging scheme counts the number of occurrences of the entities and marks these values on the corresponding entities’ first and last tokens.
- We embed the self-attention mechanism into our model to reduce the effect of irrelevant entities and highlight the crucial features.
- Our model achieves the most effective results on the NYT and WebNLG datasets. We further conduct various experiments on our model, including overlapping triples extraction and multiple triples extraction, and results show that our model exceeds baseline.

The rest of this paper is organized as follows. In section II, there is a brief overview of related work. In section III, we display our network architecture. Experiments and discussions are conducted in section IV. Finally, we conclude with a summary of our main contributions and results.

II. RELATED WORK

Most researchers treated relation extraction as two sub-tasks: NER and RC. The NER task aims to extract all entity mentions in the context. The RC task aims to recognize the relation between entities in a given text. Early works used the pipeline approach that makes two sub-tasks work independently. Such an approach suffers from error propagation since it disregards the correlation between NER and RC tasks. [3, 11, 12] proposed traditional joint models to mitigate propagation error, which need complex feature engineering and heavily rely on manual effort. The joint relation extraction model based on neural networks was studied to solve this problem and achieved state-of-the-art performance. Most neural models like [13] combined entity extraction and relation classification in a network through sharing encoder parameters. [4] first introduced a united tagging schema that can represent entity type and relation simultaneously, which converted the relation extraction task to a sequence tagging problem without identifying entity and relation separately. Previous works cannot handle the overlapping problem. To handle the overlapping problem, lots of models have been proposed and can be categorized into three classes.

The first class of works used the sequence-to-sequence (seq2seq) method to extract triples. [5] divided triples into three classes and proposed a neural model CopyRE that utilizes copy mechanism to extract triples. [14] employed a neural encoder-decoder model for extracting relations that encoder predicts one word at a time like translation machine. [15] applied reinforcement learning into the sequence-to-sequence model to handle the triples extraction order problem. However, the proposed seq2seq models hardly decode the whole span of entities.

The second class designed Multi-task learning (MTL) strategy to extract relation facts. Among these works, [16] introduced multi-task learning based on CopyRE encoding given texts with convolutional neural networks (CNN) to capture the feature of relation facts. [18] gained considerable improvement through building relation-weighted graph convolutional networks (GCN). [19] designed a novel multi-task learning architecture that enables dynamic interaction and mutual learning between NER and RC, which improves the ability to extract triples. Although effective, they lack the elegance to handle complex scenarios, such as EPO cases.

The third class method converted relation extraction to a sequence labeling problem. [6] proposed a tagging scheme based position-attention mechanism, which can solve SEO cases. [7] presented a novel decomposition strategy that hierarchically decomposes the extraction task into two sequence labeling problems but lacks the elegance to solve EPO cases. Unlike previous works, [8] proposed a new framework that maps subject to object and achieved reasonable performance in EPO cases.

Actually, these models that set an empirical value to select multiple relations inevitably lead to performance degradation. Besides, the sentences contain numerous triples in most cases. When extracting a specific triple, the model will be interfered with by other triples' feature information. As a result, these models cannot adapt to complex scenarios. Our model is based on a unified tagging scheme to extract triples. We utilize attention mechanism [20] to predict the number of relations for a given entity pair and then extract relations in a given triple.

III. METHODOLOGY

This section describes our proposed method. We first introduce the architecture of our model, where the shared encoder captures semantic features of a sentence, subject extractor recognizes subjects, and relation extractor predicts triples under a given subject. Then, we detail the novel tagging scheme of our method that converts the extraction task to the sequence labeling problem. Finally, we define the training objective.

A. Model Architecture

As shown in figure 2, our model consists of three parts: shared encoder, subject extractor (SE), and relation extractor (RE). We use the BERT [21] as a backbone to encode contextual features. The SE recognizes subjects and the RE predicts relations and objects according to these features. Formally, we extract a triple $T_j$ in the sentence $S$ and we model this process as:

$$P(T_j|S) = P(s_j|S)P(r_j|S,s_j)P(o_j|S,s_j,r_j) \quad (1)$$

where $s_j$, $r_j$ and $o_j$ represent subject, relation and object respectively in the triple $T_j$.

Eq.(1) illustrates the process of extracting triples. The first step of extracting triples in our model is to identify the
subjects according to the semantic information of sentences. The difference is that our model does not directly identify the corresponding objects or relations. In order to solve the overlapping problem, we predict a relation probability distribution table for each subject and utilize transformer encoder to predict the number of relations. Finally, our model adopts the nearest match strategy to decide the span of objects. RE takes the concatenation of the sentence representation and the hidden representation of relation probability distribution as input to improve the accuracy of prediction. Meanwhile, an attention guided layer is embedded in each module of model. Such a mechanism ensures that the module filters out noises from other irrelevant triple features.

B. Shared Encoder

The model AGFRel utilizes a pre-trained BERT model to extract feature information from a given sentence \( S = \{x_1, ..., x_n\} \), due to the excellent performance on different natural language processing tasks. BERT model employs transformer networks as the core component to obtain context-sensitive embeddings. We use WordPiece embeddings [22] to represent the words.

\[
h_i = BERT_{\text{shared}}(x_i)
\]

where \( h_i \) is hidden state at position \( i \), \( x_i \) is a one-hot vector of word indice. In the training process, we fine-tune the parameters of pre-training model to make it better adapt to relation extraction task in different scenarios. The NER and RC task use a shared encoder, which is to pass \( h_i \) into corresponding module for prediction.

C. Subject Extractor

SE aims to recognize all candidate subjects. We embed a multi-head layer to decode the vector \( h_i \). The attention mechanism allows SE to capture the interactions between two arbitrary positions and filter out the interference from other triples. Besides, the key component of BERT encoder is also self-attention mechanism, so the encoder and decoder maintain consistency.

\[
h_i^{\text{subj}} = \text{MultiHead}(h_i)
\]

\[
P(y_i^{\text{subj}}) = \sigma(W_i^{\text{subj}} \ast h_i^{\text{subj}} + b_i^{\text{subj}})
\]

\[Tag_{\text{subj}}(x_i) = \arg \max_k P(y_i^{\text{subj}} = k)\]

where \( h_i^{\text{subj}} \) denotes hidden representation of word \( x_i \) and \( P(y_i^{\text{subj}}) \) is the probability distribution of the number of subjects. \( W_i^{\text{subj}} \) and \( b_i^{\text{subj}} \) are learnable parameters of the multi-head layer. \( \sigma \) is the sigmoid activation function.

Subject extractor minimizes the sum of negative log probabilities of extracting subject candidates as below:

\[
L_{\text{subj}} = -\frac{1}{n} \sum_{i=1}^{n} \log P(y_i^{\text{subj}} = \hat{y}_i^{\text{subj}})
\]

Here, \( n \) is the length of the input sentence, \( \hat{y}_i^{\text{subj}} \) is the true tag of the \( i \)-th word.

D. Relation Extractor

RE attempts to predict relational triples \((s, r, o)\) from a sentence. Different from subject extractor, relation extractor needs to break down task into two steps. Firstly, we predict the probability distribution over the relation type \( r \) between each word and a given subject. The architecture of this component is similar to subject extractor. The specific operation is as follows:

\[
h_i^{\text{rel}} = \text{MultiHead}(h_i + h_{s+})
\]

\[
P_r(y_i^{\text{rel}}) = \sigma(W_i^{\text{rel}} \ast h_i^{\text{rel}} + b_i^{\text{rel}})
\]
where $h_{sk}$ is the $k$-th subject hidden state and $P_r(x_{rel}^{i})$ is the probability of identifying relation type $r$ between the word $x_{i}$ and the subject $s^k$. The context and subject are the main sources to support the prediction [23]. We adopt multi-head mechanism to fuse hidden representation $h_{i}$ with subject feature $h_{sk}$ into a single vector $h_{rel}^{i}$. It is worth noting that the gold subject is directly used in the training process, but not the subjects extracted by model. Secondly, we predict the number of relations between each word and a given subject as follows:

$$h_{num}^{i} = \text{Trans}(h_{i} + h_{rel}^{i})$$  \hspace{1cm} (9)$$

$$P(y_{num}^{i} = \hat{y}_{num}^{i}) = \sigma(W_{num}^{i} \ast h_{num}^{i} + b_{num}^{i})$$  \hspace{1cm} (10)$$

$$\text{Tag}_{rel}(x_{i}) = \underset{y_{rel}^{i}}{\arg \max}(P(y_{rel}^{i} = \hat{y}_{rel}^{i}))$$  \hspace{1cm} (11)$$

We denote the Transformer block as $\text{Trans}(x)$. $P(y_{num}^{i} = \hat{y}_{num}^{i})$ is the probability of identifying the number of relations between the word $x_{i}$ and the subject $s^k$. The value of $\text{Tag}_{rel}(x_{i})$ not only represents the number of relations, but also is used to recognize objects.

Relation extractor minimizes the sum of negative log probabilities of extracting relations as below:

$$L_{rel} = - \frac{1}{n} \sum_{i=1}^{n} \log P(y_{rel}^{i} = \hat{y}_{rel}^{i})$$  \hspace{1cm} (12)$$

$$- \frac{1}{n} \sum_{i=1}^{n} \log P(y_{num}^{i} = \hat{y}_{num}^{i})$$

Here, $n$ is the length of the input sentence, $\hat{y}_{rel}^{i}$ and $\hat{y}_{num}^{i}$ are the true tag of the $i$-th word. $P(*)$ is the probability of identifying true tags.

**E. Novel Tagging Scheme**

Figure 3 shows an example of how the sentences are tagged. Since identifying an entity needs to detect its start and end position, we label these critical positions with a non-zero number and the remain with 0. These labels have different meanings for different extractors. Labels in SE represent the number of subjects at their current position. For RE, labels indicate the number of relations between the object and corresponding subject. Relations can be extracted through the relation probability distribution table as shown in figure 2, in which the value of row $i$ and column $j$ means the probability that the $j$-th token is relation $i$.

For example, the word "Andrew" in start tag sequence and the word "Cuomo" in end tag sequence are both labeled 1, thus "Andrew Mark Cuomo" is a subject. When the given subject is "Andrew Mark Cuomo," RE labels the word "New" as 2, "York" as 1 and "City" as 1, which naturally solves the nested entity problem. The word "New" will be used two times to form two different triples in this sentence. The relations of two entity pairs can be inferred from the relation probability table, which are both "Born in."

Note that our tagging scheme is an improvement of ETL [7], which labels the entity’s position with entity type or relation type. This tagging method is unable to express multiple relations and nested entities. Our tagging method uses digital tags to represent these essential information, which helps model solve the nested entity problem and the overlapping problem.

**F. Training Objective**

The module SE and the module RE jointly extract triples according to our tagging scheme. The training objective of AGFRel is comprised of two components: the loss function for SE $L_{sbj}$ and the loss function for RE $L_{rel}$. We try to give different weights to the loss function of two parts and found that slightly increasing the weight of RE could make the model fit faster in the early stage. However, it does not help improve the performance of the model. The joint loss is given by:

$$L = L_{sbj} + L_{rel}$$  \hspace{1cm} (13)$$
The model minimizes the total loss $\mathcal{L}$ over all model parameters with stochastic gradient descent algorithm.

IV. EXPERIMENT

A. Datasets

We evaluate AGFRel on two public datasets NYT and WebNLG. NYT dataset was sampled from 294K articles in New York Times corpus by distant supervision method and consists of 24 predefined relation types. The WebNLG dataset was created by Natural Language Generation (NLG) tasks. We use the datasets preprocessed by [5], which contains 246 predefined relation types. It is worth mentioning that there are two methods for extracting relations. [5, 8] use partial match method simplifying an entity to the last word of an entity. [7, 16] use exact match method to extract the whole entity. [7, 16] use exact match method to extract the whole entities. For fair comparisons, we use partial match and exact match to conduct experiments. We use the preprocessed datasets released by [5]. NYT dataset contains 5000 sentences for validation and 5000 sentences for testing. WebNLG dataset contains 500 sentences for validation and 703 sentences for testing. In order to validate the effectiveness of extracting overlapping triples, the test set was divided into three parts: Normal, SEO, and EPO. The statistics are stated in Table II.

<table>
<thead>
<tr>
<th>class</th>
<th>Train</th>
<th>Valid</th>
<th>Test</th>
<th>Normal</th>
<th>SEO</th>
<th>EPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>NYT</td>
<td>56195</td>
<td>5000</td>
<td>5000</td>
<td>3266</td>
<td>1297</td>
<td>978</td>
</tr>
<tr>
<td>WebNLG</td>
<td>5019</td>
<td>500</td>
<td>703</td>
<td>246</td>
<td>457</td>
<td>26</td>
</tr>
</tbody>
</table>

B. Evaluation Metrics

We adopt the standard micro Precision (Prec.), Recall (Rec.), and F1 score to evaluate results in line with all the baselines. In the exact match task, the triples are considered correct when the whole span of entities and relations are both recognized correctly. The partial match task only needs to recognize the tail of entities and relations.

C. Implementation Details

We use Adam [24] with the learning rate $1 \times 10^{-6}$ to optimize the parameters of our model and set the batch size as 32. Dropout is applied to word embeddings and hidden states with a rate of 0.4. In this paper, we propose two models with LSTM and BERT encoder, and keep the network of decoder and encoder uniform respectively. The model with LSTM stacks two-layer BiLSTM as the encoder, one layer BiLSTM as decoder. The initial word embeddings we used are the 300 dimensions Glove [25]. The other one uses the base cased version of BERT as encoder, which contains 110M parameters. The maximum length of sentences is limited to 100 words. We implement model using Pytorch [26] on a Linux machine and train the model using Tesla V100 GPU. We choose the model that performed best on the validation set to analyze the test set.

D. Comparison Models and Results

We compare our model with three kinds of models in recent years: (1) seq2seq-based methods, including CopyRE [5] and WDec [14], (2) MLT-based methods, including GraphRel [18], CopyMTL [16] and RIN [19], (3) tagging-based methods, including NovelTagging [4], ETL-Span [7] and CasRel [8]. Table I shows the results of our models and other baseline methods. We note that our model surpasses all the baseline methods and achieves the state-of-the-art F1 score. We propose two versions of AGFRel to conduct experiments. The AGFRel$_{LSTM}$ uses the LSTM as shared encoder and replaces the multi-head guided layers as LSTM.

In the exact match task, our model AGFRel$_{BERT}$ achieves improvements of 4.7% and 4.3% in F1 scores on NYT and WebNLG datasets over the state-of-the-art models. Even with the AGFRel$_{LSTM}$ also has a relative 1.9% F1 score improvement on NYT dataset compared with MLT-based model RIN, and a relative 2.3% F1 score improvement on WebNLG dataset compared with tagging-based model ETL-Span. These results prove the effectiveness of our method.

In the partial match task, the performance of AGFRel$_{BERT}$ is close to CasRel$_{BERT}$ on WebNLG dataset. We deem that it
is because (1) the last word of the entity losses key semantic information, i.e., Apollo 12 is regarded as 12, which impedes models from extracting correct triples. (2) The performances are already saturated since the training sentences are too small for the model to learn the way to distinguish 246 relation types properly. The training sentences of NYT dataset are far more than WebNLG dataset. Our model achieves an improvement of 1.2% in F1 score. Besides, we find that CasRel$_{BERT}$ shows an imbalance between precision and recall on WebNLG. We consider that CasRel$_{BERT}$ sets a high threshold value to select relations, which sacrifices recall.

We observe that AGFRel$_{BERT}$ gains better performance on partial match task than exact match task. In the partial match task, model only needs to detect the last token of entities. We speculate that the performance gap is due to increased difficulty for the NER task. Although we mentioned that the last word of entities could not represent the original meaning of entities, entities that contains many words will also affect last word of entities could not represent the original meaning for the NER task. Although we mentioned that the last word of entities could not represent the original meaning of entities, entities that contains many words will also affect AGFRel since our tagging scheme is only to label the start and end position of entities.

E. Analysis on Different Sentence Types

To verify the ability of our model in handling the overlapping problem and extracting multiple relations, we conduct further experiments on NYT test set. We firstly divide the test set of NYT into five subclasses, each of which means a sentence contains $N$ triples. The results are shown in table III. We observe that CopyRE and GraphRel present a decreasing trend with the increasing number of triples in a sentence. Our model and CasRel$_{BERT}$ present an upward trend on the whole and are less affected by the number of triples. We attribute the difference to the reason that tagging-based methods simplify the complexity of tasks by converting relation extraction task to a sequence labeling problem. In the most challenging class ($N \geq 5$), our model achieves better performance than CasRel$_{BERT}$, since we introduce attention mechanism to eliminate the impact of irrelative triples. We also compare the results under different types of triples. AGFRel$_{BERT}$ outperforms baselines in all scenarios, which proves the validity of our tagging scheme for solving the overlapping problem.

V. Conclusion

In this paper, we propose an end-to-end sequence labeling model AGFRel for joint extraction of entities and relations based on a novel decomposition strategy. Compared to previous sequence labeling models, our model can learn the exact number of relations for each entity pair to filter out relation distracters in sentences. Our experiments show that our results exceed the baseline and achieve the optimal F1 score. Moreover, it has a good performance in handling the overlapping problem and extracting multiple triples. In the future, we will utilize GCN to encode the document-level text. The overlapping problem and nested entity problem also exist in the biomedical domain. It is beneficial to apply our method to biomedical information extraction tasks.

ACKNOWLEDGMENTS

We thank all viewers who provided thoughtful and constructive comments on this paper. This work is funded by the Fundamental Research Funds for the Central Universities. East China Normal University, Shanghai 200241, China. The experiment is completed with the support of ECNU Multifunctional Platform for Innovation (001).

REFERENCES


### Table III

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of triples</th>
<th>Overlapping pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N = 1</td>
<td>N = 2</td>
</tr>
<tr>
<td>CopyRE</td>
<td>67.1</td>
<td>58.6</td>
</tr>
<tr>
<td>GraphRel</td>
<td>71</td>
<td>61.5</td>
</tr>
<tr>
<td>CasRel$_{BERT}$</td>
<td>88.2</td>
<td>90.3</td>
</tr>
<tr>
<td>AGFRel$_{BERT}$</td>
<td>88.1</td>
<td>91.6</td>
</tr>
</tbody>
</table>

TABLE III

F1 score on sentences with different overlapping pattern and different triple count.


Multi-Label Classification of Parrott’s Emotions

Abhijit Mondal, Swapna S. Gokhale
Dept. of Computer Science & Engineering
Univ. of Connecticut, Storrs, CT 06269
{abhijit.mondal,swapna.gokhale}@uconn.edu

Abstract

Mining for latent emotions embedded in tweets can offer clues about users’ affective state on a broad range of topics ranging from their mental health to political opinions. This paper presents a multi-class supervised learning approach to group tweets into six emotions (joy, sadness, anger, fear, love, and surprise) defined according to the Parrott’s framework. After extensive pre-processing, linguistic and metadata features extracted from a corpus of tweets are used to train popular machine learning classifiers. The performance of these classifiers is evaluated using accuracy, sensitivity, and specificity computed based on a multi-class confusion matrix approach. Our framework can detect common emotions of joy and sadness with excellent accuracy (> 90%), anger and fear with moderate accuracy (75% – 85%), and love and surprise with lower accuracy (50% – 60%). Overall, the accuracy of our framework still outperforms that of contemporary approaches for all the six emotions. Further analysis of an example multi-class confusion matrix indicates that lower accuracy values for love and surprise may arise because love is often confused with joy, whereas surprise is mixed up with the positive emotion of joy and the negative emotion of fear. Moreover, this confusion could be attributed to an under-representation of these emotions in the data. This highlights the need for building high-quality, balanced benchmark data sets for training multi-label emotion classifiers.

1 Introduction

Social media platforms such as Twitter, Facebook and Instagram offer a forum for people to share and communicate with large audiences as they go about their daily lives. Twitter is one of the most popular social media platforms, with nearly 330 million monthly active users on an average as of 2019 [3]. Twitter’s large, active user base generates volumes of textual content in the form of tweets. This content shared by the users is interactive, spontaneous, conversational, and unfiltered. Tweets thus contain a treasure trove of information that can offer clues about users’ opinions, thoughts, and feelings on a variety of topics from politics to restaurants to even their mental health.

The plethora of information embedded in these tweets has attracted significant attention in their mining and analysis. A large body of work has focused on detecting and classifying the sentiment and/or polarity of the tweets [21]. In binary sentiment analysis, tweets are grouped according to positive and negative polarities, whereas in multi-class analysis they are grouped into more than two classes according to the strength of the embedded sentiment. Tweets, however, also contain affective information (moods, emotions, and feelings) of the users, and they can also be mined for these emotions. Emotion mining can thus be viewed as a deeper, more advanced form of sentiment analysis [12]. This detailed, granular information that can be extracted from tweets can support a range of applications such as targeted advertising, recommending books, music and videos, predicting the movements of stock markets, launching television programs, detecting and monitoring mental health problems, and gathering public opinion on politically and socially sensitive issues.

Emotion classification can be binary, where opposing emotions such as joy and sadness or love and hate are formulated into targeted two-way detection problems. Binary emotion classification problems can also be formulated by combining all the positive emotions such as love, joy, and trust into one class, and all the negative emotions including hate, sadness and disgust into another class. Plutchik’s wheel provides a natural anchor for formulating such two-way problems, as opposing emotions are placed on the two opposite ends of each axis on a wheel [11]. Multi-label classification of emotions, on the other hand, involves grouping tweets into many classes; these classes are usually chosen in a manner that is convenient based on the data, or are in some cases inspired by a psychological framework such as the Plutchik’s wheel [15], the Parrott’s framework [13] or...
the Ekman’s atlas of emotions [6]. Overall, in the literature, multi-label classification shows lower accuracy for all the classes or is seen to trade away the accuracy of one class for the other [14, 9, 19]. This could occur because all the emotions in a multi-class problem may not be expressed to a similar degree, that is, the data could be unbalanced. Another reason could be that these uncommon emotions are often confused or mistaken for the commonly occurring ones. To the best of our knowledge, other than the fact that multi-label emotion detection is a challenging problem, very little is known in the way of reasons behind the challenge. This objective of this paper is to present a framework that can classify a corpus of tweets into multiple emotions with good accuracy over contemporary approaches. A secondary objective is to gain deeper insights into the challenges involved in building high accuracy multi-label classifiers through a more in-depth analysis. The approach is built around a recently annotated data set [16], which tags each tweet with one of six emotions. We map these six labels to the six basic emotions defined by the Parrott’s model [13]. We extensively pre-process these tweets, extract linguistic and metadata features, and train five popular machine learning models using these features. We evaluate the performance of these models using accuracy, sensitivity, and specificity, computed based on the multi-class confusion matrix approach.

Our results indicate that the more basic and common emotions of joy and sadness can be identified with excellent accuracy (> 90%), anger and fear with moderate accuracy (75% – 85%), and love and surprise with low accuracy (50% – 60%). With these accuracy values, our classifiers still perform better than the current approaches for all the emotions. The classifiers show higher specificity compared to sensitivity, which means that they are better at ruling out a specific emotion rather than identifying it affirmatively. An analysis of an example multi-class confusion matrix indicates that love is often confused with joy, whereas surprise is mixed up with the positive emotion of joy and the negative emotion of fear, which could explain the low detection accuracy for these emotions. This confusion could occur because love and surprise are complex emotions which embody both positive and negative feelings. Moreover, because of their complexity, these emotions could be underrepresented in our corpus compared to the other classes; especially joy and sadness. Therefore, one of the ways in which the accuracy of multi-label emotion classification may be improved is by building high-quality training data sets, with a balanced representation of all the involved emotions.

The rest of the paper is organized as follows: Section 2 presents the emotion classification model. Section 3 describes the steps in the classification framework. Section 4 discusses the results. Section 5 compares and contrasts related research. Section 6 concludes the paper and offers directions for future research.

2 Emotion Classification Model

We used the data set made available by Saravia et al [16]. This set of tweets was collected using a set of hashtags, which served as noisy labels for subsequent distance-based annotations. Each tweet is labeled into one of six emotions: joy, sadness, anger, fear, love, and surprise. Annotated tweets were split into train and test data sets, with the total number of tweets in these partitions being 16000 and 2000 respectively. The number and percentage of tweets in the train/test partitions for the six emotions are summarized in Table 1. The table shows the imbalance between the emotions: joy and sadness are the most common; fear, anger and love form the next tier; whereas surprise is the most rare. However, the train/test split was conducted using stratified sampling because the ratio of test to train is maintained between 11% and 13% for all the emotions.

<table>
<thead>
<tr>
<th>Emotion</th>
<th>Train #</th>
<th>Train %</th>
<th>Test #</th>
<th>Test %</th>
<th>Test/Train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joy</td>
<td>5362</td>
<td>33.5%</td>
<td>695</td>
<td>34.75%</td>
<td>13.00%</td>
</tr>
<tr>
<td>Sadness</td>
<td>4666</td>
<td>29.17%</td>
<td>591</td>
<td>29.55%</td>
<td>12.67%</td>
</tr>
<tr>
<td>Fear</td>
<td>1937</td>
<td>12.10%</td>
<td>224</td>
<td>11.20%</td>
<td>11.57%</td>
</tr>
<tr>
<td>Anger</td>
<td>2159</td>
<td>13.50%</td>
<td>275</td>
<td>13.75%</td>
<td>12.73%</td>
</tr>
<tr>
<td>Love</td>
<td>1304</td>
<td>8.15%</td>
<td>159</td>
<td>8.00%</td>
<td>12.19%</td>
</tr>
<tr>
<td>Surprise</td>
<td>572</td>
<td>3.58%</td>
<td>66</td>
<td>3.3%</td>
<td>11.54%</td>
</tr>
</tbody>
</table>

Table 1: Summary of Tweets Per Emotion

We referred to the three most popular models of emotions to formulate the multi-label classification problem. These are the Gerrod Parrott’s model containing six basic human emotions [13], the Plutchik’s wheel of emotions [15], and the Ekman’s atlas of emotions [6]. The emotion labels in our data coincide exactly with Parrott’s model, providing us a natural anchor for our six-way classification problem.

3 Classification Framework

This section describes the classification framework.

3.1 Data Pre-processing

Our data consisted of the text of the tweets and its emotion label. It was relatively clean, and there were no emoticons, punctuations, links, hashtags and other markers. So, the pre-processing steps were relatively straightforward. First we tokenized all the words using white spaces.
In the second step, we removed the stop words using the stop words list in NLTK library [13]. In the third step, we removed proper nouns (names of persons, cities, etc.) and other non-English words by checking the presence of every word against the NLTK list.

3.2 Feature Extraction

We considered two linguistic features: Term Frequency-Inverse Document Frequency (TF-IDF) vectorization and n-grams using the bag-of-words approach. These features were extracted using the TF-IDF vectorizer class of Scikit-Learn library [2]. TF-IDF refers to a scoring measure used in information retrieval or summarization. It measures the relevance of a word in a document by assigning an additional weight to frequent words. We computed the TF-IDF scores for the topmost 1000 unigrams.

We extracted six meta-data features from the text of each tweet prior to pre-processing. These include the number of characters, number of words, number of stop words, and number of unique words, TextBlob and Vader sentiment scores. TextBlob calculates the sentiment polarity for each tweet, which ranges from $-1$ to $+1$, where $-1$, $0$ and $+1$ indicate negative, neutral and positive respectively. Vader computes a compound score as a normalized and weighted composite score obtained by analyzing each word in a tweet for its direction of sentiment – a negative (positive) valency for negative (positive) sentiment. It therefore ranges from $-1$ to $+1$ depending on the net sentiment of the tweet. We used both TextBlob and Vader scores because Vader may be more sensitive to sentiments than TextBlob, even though TextBlob may be better correlated with reviewer scores [1].

3.3 ML Models

We employed the following common machine learning models for classification. Implementations of these models in the Scikit-Learn and Keras libraries were used.

- **Random Forests (RF):** Random Forests is an ensemble learning classification technique based on decision trees. The number of decision trees is set to 30 and the number of features used by each tree is equal to the squared-root of the number of total features. Finally, each tree was allowed to grow fully up to its leaves.

- **Support Vector Machines (SVM):** Support Vector Machines is a classification method that estimates the boundary (called hyper-plane) with the maximum margin. We used SVMs with linear kernel with other default parameters.

- **Multi-Layer Perceptron (MLP):** Multi-layer Perceptron is a deep neural network that consists of input, hidden, and output layers. Our MLP model consisted of 3 hidden layers with 10, 5, and 2 neurons respectively, along with the rectifier linear unit (ReLU) activation function.

- **Gradient Boosting (GB):** Gradient Boosting is another ensemble learning classifier which builds classifier trees such that each tree takes a small step towards the minimization of classification error from the previous tree. The algorithm continues until maximum number of trees are built or there is no significant improvement in minimizing the error. Finally, predictions for the test data are obtained by combining predictions of the trees built in each stage using a weighted sum. We used 100 estimators, with a maximum depth of 1.

- **Neural Network (NN):** We build a neural network with three layers having 30, 10 and 6 neurons respectively. We arrived at this architecture through experimentation, considering that our data was of medium complexity with about 1000 features.

3.4 Performance Metrics

For multi-label classification, the first step in defining the performance metrics is the computation of the multi-class confusion matrix, which represents how many of the tweets originally in that class are classified accurately as belonging to that class. Also, for a given class it represents the number of tweets that are mis-labeled by a classifier as belonging to each of the other five classes. Finally, we divided each of these six counts by their sum to obtain a normalized accuracy measure. For example, let 500 tweets be originally labelled as “surprise”. Now, suppose if 400 of these tweets were labelled correctly by the classifier as “surprise” but the other 100 tweets were mis-labelled. Further, suppose that these 100 tweets were mis-labelled equally among the other five classes meaning each of the other five classes included 20 of these tweets. Next, we divide these six counts by 500 to compute the six elements in the normalized multi-class confusion matrix as 0.8, 0.04, 0.04, 0.04, 0.04, and 0.04. We repeat this process to calculate all 36 entries in the confusion matrix. The accuracy for each class is defined as the percentage of tweets labeled correctly from that class, and refers to the diagonal elements in the confusion matrix.

Alongside multi-label classification accuracy, we also calculated two other performance metrics, namely, sensitivity and specificity. Sensitivity and specificity together offer insights into the bias of a classifier towards a particular class. However, these two performance metrics are mainly used in the context of binary classification problems, as they need us to define positive and negative classes in order to be able to compute true and false positives, and true and false
For example, to calculate the sensitivity and specificity for surprise, we considered the “surprise vs rest” classification problem. We designated the positive class as “surprise” and all the other classes together formed the negative class. True positives (TP) are the tweets which are correctly classified as “surprise”, true negatives (TN) include tweets originally not from the “surprise” class and are also not labeled as “surprise” by a classifier. Similarly we can define false positives (FP) as those tweets that were incorrectly labeled as “surprise”, and false negatives (FN) as those tweets that were originally labeled as “surprise” but the classifier labeled them incorrectly with one of the other five classes.

Equation (1) shows the expressions for sensitivity and specificity. Sensitivity of “surprise” class is the percentage that a tweet labeled as “surprise” is correctly classified as such. We note that sensitivity is identical to multi-label accuracy. If a highly sensitive classifier classifies a tweet into an emotion class, then it can be fairly certain that it actually does. Specificity of “surprise” class is the percentage that a tweet which is not labeled as “surprise” is classified as such. If a highly specific classifier says that the tweet does not exhibit an emotion, then we can be fairly certain that it indeed does not. Generally, there is a trade-off between sensitivity and specificity. A classifier with a high sensitivity usually has low specificity, and vice versa.

After computing the sensitivity/accuracy and specificity for each emotion, we compute the aggregate unweighted and weighted values of these metrics across all classes. The weight for each class is given by the percentage of tweets in that class in the training data set.

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \\
\text{Specificity} = \frac{TN}{TN + FP}
\] (1)

4 Results & Discussion

The data was already split into train and test sets for model training and performance evaluation. Table 2 shows the per-class accuracy for the five models. With our unweighted and weighted overall accuracy values of 74% and 83% for the NN model, our approach outperforms many contemporary approaches with accuracy values in the ranges of 50 - 60% [4][9][9]. Tables 2 and 3 summarize the sensitivity/accuracy and specificity values for the six classes by the five classifiers. Across all the emotions, sensitivity values are lower than their corresponding specificity values. All specificity values are around or over 95%, meaning that the classifiers are excellent at negative differentiation, that is, they can identify with near certainty, the absence of a specific emotion. Lower sensitivity values indicate that the classifiers are less capable of zeroing in on a specific emotion. The classifiers can identify joy and sadness with excellent accuracy, fear and anger with moderate to low moderate accuracy, but struggle with love and surprise; more so with surprise than with love. Albeit low, our accuracy in detecting surprise still exceeds the accuracy of the contemporary works that have simply been unable to detect this emotion [14]. Neural networks offer the best specificity across all the emotions. Sensitivity produces mixed results among models; for each emotion the best model for sensitivity is different and is identified in the parentheses: joy (SVM), sadness and fear (NN), anger and love (MLP), and surprise (RF). Generally, the difference in sensitivity/accuracy between the models is small for all emotions except for surprise, where MLP diverges significantly.

To understand why the classifiers may struggle with the emotions of love and surprise, we take a closer look at an example multi-class confusion matrix from the SVM model (matrices from other models show similar trends) as shown in Figure 1. This matrix shows how the tweets from each class are mis-classified into the other five classes. From the figure, it can be seen that love is most likely to be confused with joy while surprise is most likely to be confused with either joy or fear. Therefore, the expression of love is almost always positive, whereas, surprise can be expressed in both positive and negative senses; and it embodies both these emotions. This confusion, which leads to lower accuracy for love and surprise could be due to class imbalance; Table 1 shows that only 8% tweets are labeled as love, and an even lower 3% tweets are labeled as surprise.

Figure 1: Multiclass confusion matrix (SVM)
5 Related Research

Prevalent research efforts have mined emotions surrounding specific events such as the presidential election [20] or the Brazilian soccer league [5], or natural disasters such as the California Camp Fire [10] and the MERS outbreak [4]. However, extracting them from a general corpus remains relatively unaddressed.

Many research works formulate multi-label classification problems over a set of emotions; the chosen set may be completely ad hoc, inspired by a psychological framework such as the Ekman’s atlas of emotions [6] or the Plutchik’s wheel [15], or a combination of psychology and heuristics. For example, Wang et. al. [19] annotated a data set of 2.5 million tweets based on hashtags related to emotion words, and classified them into seven emotions, six basic plus “thankfulness”. Their classification accuracy is around 60%, and this performance is further improved by about 5% [9]. Jaishree et. al. [14] label tweets by combining the scores from NRC word-level lexicon tool and emotion-based hashtags. Their problem considered 8 basic emotions on the Plutchik’s wheel, however, their multi-label classification problem was completely unable to detect surprise, and registered low scores for fear. A smaller set of 4 emotions is also used by some [7, 17]. Although Mohammed et. al. formulate their problem based on the Plutchik’s wheel, they ultimately boil it down to binary classification by using the one vs. other method [8]. Generally, multi-label emotion classification suffers from either low accuracy for all classes or sacrifice the accuracy of some for the others. The accuracy values of our approach are higher for all the emotions compared to these contemporary approaches. Moreover, a detailed analysis sheds further light into those emotions that are difficult to detect, and how they could be confused with the others.

6 Conclusions and Future Research

Simultaneous differentiation between multiple emotions from content shared on social media platforms remains a challenging problem. This paper proposes a classification framework based on supervised machine learning that can identify six emotions of joy, sadness, anger, fear, love, and surprise defined in the Parrott’s framework from a corpus of tweets. Relying on extensive pre-processing of tweets, followed by the extraction of linguistic and metadata features to train popular machine learning models, our classification framework can identify joy and sadness with excellent accuracy, anger and fear with moderate accuracy, and love and surprise with low accuracy. Moreover, the aggregate accuracy of our approach is better than contemporary approaches. Through a detailed analysis, we develop insights into why love and surprise could be difficult to detect, and offer that one plausible explanation for this difficulty could stem from an under-representation of these two emotions in the data.

Our future research involves building a high-quality balanced data set that can be used to train classifiers for multi-label emotion classification. Experimenting with identifying emotions surrounding high profile events related to Covid-19 such as vaccinations, or the passage of the American Rescue Plan Act of 2021 is also a topic of the future.
References


On Conducting Tests in Software Engineering Courses during the COVID-19 Pandemic

Pankaj Kamthan
Computer Science and Software Engineering
Concordia University
Montreal, Canada
pankaj.kamthan@concordia.ca

Abstract—The circumstances presented by the COVID-19 pandemic have had a severely debilitating impact on education, in general, and software engineering education, in particular. This paper describes the author’s experience over three semesters in conducting oral and written tests in six software engineering-related courses with around 500 students overall. The technical as well as non-technical challenges encountered are discussed, and educational lessons based on the reactions and responses of students are given. The proposed concepts, decisions, and processes herein are effectively generalizable and potentially applicable to other types of courses and to (not necessarily pandemic-related) online teaching-related situations in general.

Keywords—collusion and plagiarism; conceptual model; distance education; e-learning; formative and summative assessment; online teaching and learning

I. INTRODUCTION

The COVID-19 pandemic was an unprecedented, disruptive event of 2020 that continues to have severe ramifications on a global scale in 2021. To prevent infections, to mitigate and control the rate of transmission, and to minimize hospitalizations, the governments everywhere have been compelled to declare community lockdowns, and to closing or restricting access to all those sectors of society that necessitate a large number of people congregating in closed spaces for a prolonged period of time, including educational institutions [1]. In order to continue to remain operational, the educational institutions have inexorably resorted to a suitable combination of e-learning, distance education, and online teaching and learning [2]. This paper reports the experience at a University of conducting tests in six courses over three semesters, Spring 2020, Summer 2020, and Fall 2020, in such an environment. These courses are part of an undergraduate program (namely, Introduction to Software Engineering and Object-Oriented Methods with UML), and a graduate program (namely, Advanced Software Architectures, Software Comprehension and Maintenance, Software Engineering Development Processes, and Software Measurement), topics of which are aligned with a taxonomy of software engineering education [3].

The rest of the paper is organized as follows. In Section II, relevant background is provided and related work is discussed. The specifics of the tests, along with associated challenges and lessons learned are presented in Section III. In Section IV, directions for future research are outlined briefly. Finally, in Section V, concluding remarks are given.

DOI reference number: 10.18293/SEKE2021-025

II. BACKGROUND AND RELATED WORK

The COVID-19 pandemic has obligated teachers from around the world, at every level, in all disciplines, to improvise and innovate in order to continue to provide an acceptable level of education. The notions of e-learning, distance education, and online teaching and learning have existed for quite some time. Indeed, they have been practiced to varying degrees over the years, especially since the inception of the Web, and their advantages and disadvantages are known [4, 5]. However, they have garnered special attention, assumed renewed significance, and seen a broad adoption during the COVID-19 pandemic.

The sudden transition from offline to online teaching has led to, somewhat expectedly, mixed results for administrators, teachers, and students [6, 7, 8, 9]. In [10], the experience, both before and after the declaration of COVID-19 as a pandemic, of conducting tests using MyElearning in a digital electronics course is described. In [11], the experience of conducting a test comprising multiple choice questions and a case study-based question using Google Forms, Google Drive, and WhatsApp in a multimedia systems course has been given. In [12], issues related to conducting unproctored online tests using a question bank and Examplify in a general chemistry course are explored. There have been relatively few detailed reports specific to test procedures, processes, and practices during the COVID-19 pandemic, and that, in part, forms the motivation for this paper.

III. THE TESTS

The courses in question had a number of components of assessment, including tests. The tests were conducted in alignment with the principles of online teaching [13].

A. Students

In some cases, the students were geographically distributed across different cities within the same country (Canada) and, in other cases, across different cities in different countries (Canada, China, France, India, and Morocco). In order to prepare the students for oral and written tests, they were (1) told that the standard of questions and problems would be same as in an offline teaching environment, and (2) given sample questions/answers and problems/solutions, as the case may be, well in advance so that they could practice and familiarize themselves with the unique style of the upcoming tests. The students were either accustomed to the technologies and tools used for the tests, or learned them readily on their own.
B. Technology and Tool Environment

The technologies and tools for the tests were provided by the University, and were deemed suitable for online tests [14]. The syllabus, schedule, and other pertinent details of the tests were announced on Moodle, a course management system. The tests were conducted in real-time using Zoom, a cloud-based video communications software-as-a-service (SaaS). The students submitted their written tests on a special-purpose electronic assignment submission system (EAS) supported by the University. The access of each of these systems required a username-password-based authentication. The students were given the alternative to submit their tests via e-mailing the teacher in the event that a submission on the EAS was not possible for technical reasons (say, if the system was unavailable because it was overloaded or was non-operational).

C. Oral Tests

The COVID-19-related regulations of the University were such that they allowed the students to attend the lectures without using any audio or video on their side. (The students were given the right to turn their camera and/or microphone off during the lectures, and many did exercise that right.) The purpose of the oral test was, therefore, not only about an assessment of their verbal aptitude or ability to answer technical questions in a relatively short period of time, but also to improve personal rapport between the teacher and the students, by being able to see them, speak to them, and hear them 'in person', in real-time.

1) Description of the Prerequisites, Regulations, and the Process of Oral Tests

There was one oral test per course. The duration of the test, depending on the class size, lasted between 2 hours to 8 hours. A pool of questions was prepared in advance of the test. To reduce repetition and to decrease the possibility of students who were tested earlier to help those who were tested later, the number of questions exceeded the class size. The questions were either open or closed, but mostly closed for which answers were also prepared in advance of the test. The teaching assistants were consulted to ensure that the questions were within scope of the syllabus and answerable within the time allowed. The examiners during the test included the teacher and all the teaching assistants for that course. Each student was given a specific time slot of 10-minute duration. The students were prohibited from entering a time slot not allotted to them, given a specific time slot of 10-minute duration. The students in the same time slot. The examiners rotated in asking questions. The initiator of a question (namely, the teacher or one of the teaching assistants) was pseudo-random (that is, not predetermined). The students were not allowed to revisit a question that they could not answer earlier. This was done to prevent students seeking external help, such as asking someone or checking the Web for a possible answer. The session was recorded in case a student may wish to revisit answers that the student provided, and some students did. The marks were determined and recorded in real-time by the examiners.

2) Challenges and Lessons Learned from Oral Tests

- **Scope of Questions.** The questions did not include any that would require the need for calculating, diagramming, or programming. In case of a question with an open answer, some students attempted to give a rather lengthy, albeit correct, answer. For consideration of time, such students had to be interrupted and were told to stop speaking to allow sufficient time for other questions (and/or for other students to have their turn to be asked).

- **Understanding Questions.** In some cases, the students had difficulty listening to a question, perhaps due to technical (audio) problems or due to the way it was read and spoken by an examiner. In such cases, the question was repeated, verbatim or with a slight variation.

- **Quality of Responses.** The quality of responses varied considerably among the students. There were primarily three types of problems: (1) incorrect or partially incorrect answer, (2) answer not matching the question, and (3) over-answering (correct answer that was annotated with extraneous statements that were not always correct). It is unclear whether those students whose verbal responses was deemed problematic would have said or done anything different if they had more time, if were allowed to express their answers in writing, or if they were in an offline teaching environment.

- **Student Anxiety.** There were some students who had never been in an interview or given an oral test previously,
and were therefore noticeably, as their facial expression and/or body language implied, unusually anxious.

- **Duration of Test.** The duration of the test was not an issue for the students, but was somewhat exhausting for the examiners in those cases where it exceeded 4 hours.

- **Collusion and Plagiarism.** In spite of the steps taken to prevent collusion or plagiarism, it did appear from the nature and/or the promptness of the answers to non-trivial questions that some students either attempted to collude/plagiarize, or may even have colluded/plagiarized successfully. (For example, in a few cases, the students appear to look elsewhere for an unusually long period of time as if they were staring at another person or at a different screen.) However, in absence of concrete evidence, no cases were pursued. In case of a suspicion, the responses given by the students to the original question were dismissed, and they were asked an ‘extra question’ after being reminded to look directly at the camera at all times while answering.

### D. Written Tests

1) **Description of the Prerequisites, Regulations, and the Process of Written Tests**

There were two written tests per course. The tests assumed that the students had access to (at least) the lecture notes, a calculator, a dictionary, Zoom, and the ability to upload documents to the EAS. The students were provided with a Microsoft Word or LATEX template, depending on the course, and were expected to give a solution to each problem in the template provided. The students were told that the problems would be based strictly on the syllabus specified, and, to discourage the use of sources that were not permitted on the test, the solutions that included claims based on querying a public search engine would not get credit. To encourage the students to make proper use of the time allowed, they were also told to place attention on quality, not quantity, in writing, and that, in general, they should avoid overwriting. The students were also told that for each problem, certain space and certain time were allocated, and that they should manage both space and time. The students were told that, in general, they were not to communicate, including seek clarification on a problem, as the ability to read and understand the problems given was part of the test. However, in case of an emergency, they could communicate privately with the teacher either by e-mail or by using the chat feature of Zoom. Finally, the students were told that it would be in their interest to have their technical environment ready in advance of the test. This included Zoom running, Microsoft Word template open, and the EAS URL in the Web user agent entered before the test commenced.

The process for the written test is shown in Fig. 2 and Fig. 3 as a collection of UML Activity Diagrams from the perspectives of the examiner and the student, respectively. In Fig. 2, T is the time allocated for a problem on a particular page/screen. The URL for a rendezvous on Zoom was announced privately (rather than publicly, for security reasons) on Moodle. The tests had 20 to 24 problems each, depending on the course. In each test, every problem was allocated a certain amount of time, which ranged from 2 to 6 minutes. The tests proceeded as follows. At the start of a test, a screen rendering a PDF document was shown with a problem statement and the duration for that problem. There was one problem per page (and, therefore, per screen). For each problem, the students were given a 1-minute verbal reminder before the duration of that problem lapsed. Upon the lapse of duration for a given problem, the screen was shifted to the next problem, in a sequential manner. There was no backtracking, meaning it was not possible for the students to go back to any of the problems shown previously. At the end of the test, the students were to upload their tests on the EAS, and everyone did. There was per-minute penalty for a late submission, regardless of the reason, and some students were penalized.

The tests were not proctored, despite the possibility. The students were not required to have their camera on, and none of them did. To avoid disturbing the others taking the test, the students were also told not to use their microphone under any circumstance, and, again, none of them did.

- **Figure 2.** A UML Activity diagram for the written test process from an examiner’s perspective.

- **Figure 3.** A UML Activity diagram for the written test process from a student’s perspective.
2) Challenges and Lessons Learned from Written Tests

- **Problems Requiring Mathematical Notation.** The problems required solutions with only simple and short mathematical expressions, if at all, as typesetting mathematics can be tedious and rather time consuming.

- **Quality of Writing.** The quality of writing varied considerably among the students. It is unclear whether those students whose writing was deemed problematic would have done anything different if they had more time or if they were in an offline teaching environment.

- **Quality of Diagramming.** The students were informed in advance that they may need to draw simple UML Class Diagrams, UML Component Diagrams, and/or UML Use Case Diagrams. The syntactic, semantic, and/or pragmatic quality of the diagrams varied, primarily because of the time allocated to problems that required diagramming was, as self-reported by the students, deemed insufficient.

- **Weak Internet Connection.** There were a few students who were on a Wi-Fi connection that, as self-reported by those students, turned out to be not reliable enough for the tests. These students had to repeatedly reconnect to Zoom.

- **Students with Disabilities.** The University regulations require that the students who are registered with the Access Centre for Students with Disabilities (ACSD) be given special accommodation during the tests. In particular, the students registered with the ACSD are, in some cases, given extra time for the tests, where the exact extra time is on a per case basis, depending on the student. In some of the courses, there were students registered with the ACSD, and, for such students, the test process was repeated with a longer than usual duration.

- **Collusion and Plagiarism.** In spite of the steps taken to prevent collusion and plagiarism, there were a few cases where, as the evidence suggested clearly, some students got together in small groups and copied from each other. These cases were duly identified, pursued, and reported.

**E. Discussion**

The students were solicited to provide feedback on their experiences, with the tests and otherwise, in the courses. There were no objections from the students regarding the fairness of any of the tests. The nature of the types of problems or the differences in time zones did not seem to have any notable impact on the performances of the students in any of the tests.

IV. DIRECTIONS FOR FUTURE RESEARCH

There are several established ways of, as well as technical environments for, conducting an online test [4, 14]. It could be useful to analyze and compare the degree of comfortableness, individual as well as communal behavior, performance, and the extent of learning of students in written tests as described in this paper with the same or similar group of students (1) in an offline teaching environment, (2) using a different style of question/answer or problem/solution methodology, and/or (3) using a different set of technologies and tools, and is therefore of research interest.

V. CONCLUSION

The obstacles presented by the COVID-19 pandemic to (software engineering) education go beyond that of those typical of ‘conventional’ distance education or online teaching and learning. However, this paper has shown that using only basic technologies and tools, it is possible to conduct verbal and written tests, with acceptable outcomes, albeit with associated challenges, some of which could, hopefully, be overcome in due course with better planning and preparation.

**ACKNOWLEDGMENT**

This paper is dedicated to the indelible memories of Terrill Fancott and Peter Grogono, who passed away in 2020 and 2021, respectively. They inspired and guided the author over the years in many matters related to software engineering education. The author is grateful to CUPFA for a Professional Development Grant and to his teaching assistants for useful discussions.

**REFERENCES**


Revisiting UML Class Relationship Recovery for Online Education

Dionysis Athanasopoulos
School of Electronics, Electrical Engineering, and Computer Science
Queen’s University of Belfast, Northern Ireland, UK
D.Athanasopoulos@qub.ac.uk

Abstract

UML recovery has been a long-standing challenge for the software-engineering community. The complete recovery of UML class relationships needs the employment of both static and dynamic code analyses. However, the dynamic-code analysis is not usually applicable at the design time of programs and especially for incomplete programs in online education. To overcome this restriction, we propose a formally defined set of mappings between UML relationships and object-oriented relationships that are based on static-code analysis exclusively. We evaluate the precision and the recall of our mappings on student projects against ground-truth UML diagrams and against diagrams recovered by existing UML class recovery tools.

1 Introduction

The motivation of our research come from a real story. It all started a few weeks ago in the labs of a computer-science school. Amelia\(^1\), an undergraduate student, wanted in the context of a software-design module to take online feedback on her UML class diagrams\(^2\) that visualize the design of her Object-Oriented (OO) programs. Class diagrams describe the static structure of OO programs by showing the program’s classes, fields, methods, and class relationships.

Amelia generally feels confident to build up a UML diagram only if she can map it to the source-code elements that implement the diagram. In other words, she prefers first writing (a skeleton of) her OO programs and then mapping them to UML diagrams via using her favorite integrated development environment, IDE (e.g., Eclipse\(^3\)). The program that she has started developing today contains classes that are related to each other in various ways. Amelia found it difficult to build the diagram on her own and especially, to differentiate the usage of the various kinds of arrows that UML provides. In particular, she was confused while she was mapping the implementation-level relationships of her program to UML class arrows.

Thus, Amelia needed an online tool that takes as input her OO programs and outputs a visual medium for her programs. Such a tool should be quite precise with respect to the usage of the UML arrows. Moreover, the tool should be able to work on incomplete programs that cannot necessarily be executed. In other words, the tool should be based on the static-code analysis of OO programs. Amelia thought such a tool is a necessary classroom assistant in the era of online education that has recently stressed.

Luckily for Amelia, her module owner, Bob, suggested to her to use a freeware (e.g., ObjectAid\(^4\)) that can be integrated with her IDE and recover UML diagrams from incomplete Java programs. Amelia was happy to see that the tool can draw UML diagrams by just dragging and dropping Java classes, providing a visual medium for Java programs. However, when she used the tool for her programs, she was concerned about the arrows used by the tool used for some Java class relationships in the recovered diagram. To double check the diagram, Amelia discussed her concerns with the module owner. Bob drew his own diagram and verified Amelia’s concerns about the precision of the tool, as analysed in a next section of the current paper.

Overall, existing UML recovery tools that use static-code analysis are not precise enough for online learning purposes. Moreover, the state-of-the-art research approaches that could be adopted for overcoming this limitation are not completely based on static-code analysis (e.g., \([1, 2]\)) or they do not satisfy the lifetime and the share-ability object properties required for recovering the UML composition relationship (e.g., \([3]\)).

We contribute an initial version of an automatic approach that takes as input an OO program and outputs the expected UML class relationships. To this end, we formally define the concepts of OO classifier, OO relationship, and UML relationship via using static-code syntactic analysis exclu-
We further propose a formally defined set of mappings between OO and UML relationships that satisfy the required lifetime and the share-ability properties. We finally evaluate the precision and the recall of our mappings on existing student projects against ground-truth UML diagrams and against diagrams recovered by existing professional UML class recovery tools.

The rest of the paper is structured as follows. Section 2 presents the related research approaches. Section 3 defines the concept of OO relationships. Section 4 maps OO relationships to UML relationships. Section 5 evaluates the effectiveness of our approach. Finally, Section 6 summarizes our contribution and proposes future research directions.

2 Related Work

UML class diagrams represent OO classifiers (e.g., class, interface), fields, methods, and classifier relationships. The UML standard defines the following kinds of relationships between classifiers: dependency, inheritance, realization, association, aggregation, and composition. The association between classifiers is a directed or a unidirectional relationship.

We organize the existing approaches of the round-trip engineering between UML and OO programs into three categories. The first-category approaches generate source code from UML diagrams based on UML to OO mappings. Other approaches recover business processes from UML sequence diagrams by using a set of heuristics. The second-category approaches define consistency links between UML diagrams and source code.

The third category includes reverse-engineering approaches that recover (parts of) UML diagrams from OO source code. recovers UML use-case diagrams by using trace-ability links between use-case elements and classifiers. recovers UML behaviour diagrams from source code by identifying patterns in the source code. recover UML relationships by identifying mappings between UML and OO relationships.

Our approach belongs to the third category and is related to [1, 2, 3]. recover UML relationships via checking the following set of properties for objects: multiplicity, exclusivity, and lifetime. However, static and dynamic code analyses are used to confirm the properties.

recover composition relationships via checking the non-accessibility property for objects. To this end, check whether a reference to an object is exported by its owner object to a third-party object. However, states that the definition of composition based on the non-accessibility property is not consistent with the UML specification. further states that the lifetime and the share-ability properties are the properties that should be used for recognizing composition relationships. specifies an OCL formalization of the above properties. However, the complete verification of the above properties needs both static and dynamic code analyses.

Program 1 OO Skeleton of the Flight-Booking Program

3 Object-Oriented Relationships

We illustrate our definitions via using a running example. We take an example that corresponds to a small part of an OO flight-booking system. The program calculates the total price of a booking and prints out the overall booking information. The Java-like pseudo-code of the classes of the above program is provided in Prog. 1.

Classifier fields and methods. A classifier mainly consists of classifier-level fields (e.g., built-in data-types, objects of other classifiers) and/or methods.

Owned object reference. An classifier can be associated to an object of a classifier even if has not created the
A classifier, \( c \), includes (i) its name \( n \) whose prefix is its package path (this combination can uniquely identify the classifier in a program); (ii) its kind \( k \) (concrete class, abstract class, interface, enum); (iii) a (possibly empty) set of generic classifiers \( g_i \) that \( c \) extends/implements; (iv) a (possibly empty) set of classifier-level developer-defined \( f_i \) fields (owned objects), along with their maximum \( l_i \) multiplicity; (v) a (possibly empty) set of the \( d_i \) classifiers whose references are explicitly finalized by \( c \); (vi) a (possibly empty) set of the \( r_j \) object references that are owned by \( c \); (vii) a (possibly empty) set of the methods of \( c \); (viii) a (possibly empty) set of the \( u_i \) classifiers (along with their maximum \( l_i \) multiplicity) whose objects are created by \( c \). If the object is created by using a combination of generic and concrete classifiers, then the \( u_i \) set includes both the generic and the concrete classifiers.

\[
c = (n, k, \{g_i\}, \{(f_i, l_i)\}, \{d_i\}, \{r_i\}, \{m_i\}, \{(u_i, l_i)\})
\]

Definition 2 (Method) A method is characterized by (i) its name \( n \); (ii) a (possibly empty) set of \( \arg \) arguments that are developer-defined classifiers (along with their maximum \( l_i \) multiplicity); (iii) its (possibly absent) return developer-defined return type (along with its maximum \( l_i \) multiplicity).

\[
m = (n, \{(\arg, l_i)\}, \{\text{ret,} \ l\})
\]

Definition 3 (Owned Object) A \( c_1 \) classifier owns an object of a \( c_2 \) classifier if the \( c_2 \) object belongs to the developer-defined fields of \( c_1 \) and the \( c_2 \) object has been created by the \( c_1 \) classifier. To put it formally, a \( c_2 \) object is owned by \( c_1 \) if the following condition is evaluated as true.

\[
\text{ownedObj}(c_2, c_1) := c_2 \in c_1.\{f_i\} \land c_2 \notin c_1.\{u_j\}
\]

Definition 4 (Owned Reference) A \( c_1 \) classifier just owns a reference to an object of a \( c_2 \) classifier if the \( c_2 \) object reference belongs to the developer-defined fields of \( c_1 \) but the \( c_2 \) object has not been created by the \( c_1 \) classifier.

\[
\text{ownedRef}(c_2, c_1) := c_2 \in c_1.\{f_i\} \land c_2 \notin c_1.\{u_j\}
\]

Definition 5 (Associated Reference) A \( c_1 \) classifier is associated with a reference to an object of a \( c_2 \) classifier if the \( c_2 \) reference does not belong to the developer-defined fields of \( c_1 \), the \( c_2 \) object has not been created by the \( c_1 \) classifier, and the \( c_2 \) object is included in the arguments of a method of the \( c_1 \) classifier.

\[
\text{assocRef}(c_2, c_1) := c_2 \notin c_1.\{f_i\} \land c_2 \notin c_1.\{u_j\} \land c_2 \in c_1.m_k.\{\arg\}
\]

4 \textbf{OO and UML Relationship Mapping}

According to [15], composition should be defined based on the lifetime and the share-ability properties. The share-ability property requires that an object of a classifier, along with the references to the object, must be owned by at most one composite classifier. The lifetime property requires that the object of a composite classifier cannot be outlived by its owned objects. In other words, when the object of a composite classifier is finalized, its owned objects and the references to the owned objects are finalized too.

Definition 6 (Object Share-ability) A \( c_1 \) classifier shares a \( c_2 \) object with a \( c_3 \) classifier if there is a reference owned by \( c_3 \) to the \( c_2 \) object that is created and owned by \( c_1 \).

\[
\text{share}(c_1, c_2, c_3) := \text{ownedObj}(c_2, c_1) \land \text{ownedRef}(c_2, c_3)
\]

To compare the lifetime between an object of a composite classifier and its owned objects via using OO relationships, we define and prove the following theorem that is based on the object share-ability.

Theorem 1 (Composite object lifetime) The lifetime of an object of a \( c_1 \) composite classifier is longer than or the same to the lifetime of an object of a \( c_2 \) classifier that is owned by \( c_1 \) if there is no other \( c_3 \) classifier that explicitly finalizes the \( c_2 \) object and \( c_3 \) does not own a reference to the \( c_2 \) object. If \( c_3 \) owns a reference to the \( c_2 \) object, then \( c_1 \) should explicitly finalize \( c_2 \).

\[
\text{life}(c_1, c_2) := \text{ownedObj}(c_2, c_1) \land c_2 \notin c_3.\{d_i\}
\]

Proof 1 We assume that a \( c_2 \) object is owned by a \( c_1 \) object and we examine all the possible cases with respect to the ownership of the \( c_2 \) object/references and the finalization time of the objects.
(a) If \( c_3 \) explicitly finalizes \( c_2 \), then \( c_1 \) cannot use its owned object \( c_2 \) and consequently, the lifetime comparison of \( c_1 \) and \( c_2 \) is meaningless (the second condition is false).

(b) If \( c_3 \) does not explicitly finalize \( c_2 \), \( c_2 \) owns a reference to \( c_2 \), and

\[
\begin{align*}
(i) & \text{ } c_1 \text{ is finalized without finalizing } c_2 \text{ (swallow finalization), then } c_2 \text{ has longer lifetime than } c_1 \text{ because there is a live reference to } c_2 \text{ in the } c_3 \text{ object (both third and fourth conditions are false)} \\
(ii) & \text{ } c_1 \text{ and } c_2 \text{ are finalized together (deep finalization), then } c_1 \text{ and } c_2 \text{ have the same lifetime and } c_3 \text{ cannot use } c_2 \text{ because } c_2 \text{ has been finalized (the first, second, and fourth conditions are true)}
\end{align*}
\]

(c) If \( c_3 \) does not explicitly finalize \( c_2 \), if there is no \( c_3 \) object that owns reference(s) to \( c_2 \) and

\[
\begin{align*}
(i) & \text{ } c_1 \text{ is finalized without finalizing } c_2 \text{ (swallow finalization), then there is no left object that uses } c_2 \text{ and we consider that the lifetime of } c_1 \text{ and } c_2 \text{ is the same (the first, second, and third conditions are true)} \\
(ii) & \text{ } c_1 \text{ and } c_2 \text{ are finalized together (deep finalization), then } c_1 \text{ and } c_2 \text{ have the same lifetime (all conditions are true)}
\end{align*}
\]

Illustrative example. The Flight object owns a Business object in Prog. 1, but there is no reference to the same Business object owned by another object. According to Theorem 1, the lifetime of the Flight object is longer or the same to the lifetime of the Business object.

Definition 7 (Composition) A \( c_1 \) classifier is composed by a \( c_2 \) classifier if there is no \( c_3 \) classifier that shares with the \( c_1 \) classifier the same \( c_2 \) object and the lifetime of the \( c_1 \) object is longer or the same to the lifetime of the \( c_3 \) object.

\[
\text{comp}(c_1, c_2) := \exists c_3 : \text{share}(c_1, c_2, c_3) \land \text{life}(c_1, c_2)
\]

Illustrative example. The Flight object in Prog. 1 owns a Business object, there is no reference to the same Business object that is owned by another object, and the Flight and the Business objects have the same lifetime. In this case, the Flight and the Business classes have a UML composition relationship.

Aggregation relates a composite classifier and its owned objects/references. To capture this relationship, we use the owned object and reference relationships (Def. 3 and Def. 4), without the composite and the owned objects/references satisfying the lifetime and the share-ability properties.

Definition 8 (Aggregation) \( c_1 \) classifier aggregates a \( c_2 \) classifier if \( c_1 \) owns \( c_2 \) object(s)/reference(s) but \( c_1 \) does not have a composition relationship with \( c_2 \).

\[
\text{aggr}(c_1, c_2) := (\text{ownedObj}(c_1, c_2) \lor \text{ownedRef}(c_1, c_2)) \land \neg \text{comp}(c_1, c_2)
\]

According to the UML standard, association exists when a classifier is associated with references to object(s) of another classifier. In other words, the association can be defined by using Def. 4. But if the former classifier is composite that owns the object(s)/reference(s) of the latter classifier, then the classifiers may have a composition/aggregation relationship.

Definition 9 (Association) \( c_1 \) classifier is associated with a \( c_2 \) classifier if \( c_1 \) object does not own a \( c_2 \) object/reference and the \( c_1 \) object is associated with a reference to a \( c_2 \) object: \( \text{assoc}(c_1, c_2) := \text{assocRef}(c_1, c_2) \)

Please note a set of binary associations can be combined to form N-ary associations that may exist. However, the current work focuses on the recovery of binary associations, leaving as future work the recovery of N-ary associations.

Definition 10 (Realization) \( c_1 \) classifier realizes a \( c_2 \) classifier if \( c_2 \) is an interface and \( c_1 \) implements \( c_2 \).

\[
\text{impl}(c_1, c_2) := c_2 = c_1.g_i \land c_2.k = \text{"interface"}
\]

Definition 11 (Inheritance) \( c_1 \) classifier inherits from a \( c_2 \) classifier if \( c_1 \) extends \( c_2 \) and \( c_2 \) is concrete/abstract class: \( \text{inher}(c_1, c_2) := c_2 = c_1.g_i \land c_2.k = \text{"class"} \)

The dependency generally indicates that a source classifier uses an object of a target classifier. But if the former is a composite classifier that owns the object(s)/reference(s) of the latter, then the classifiers may have a composition/aggregation relationship. Otherwise, if the former uses a reference to an object of the latter, then the classifiers have an association relationship.

Definition 12 (Dependency) \( c_1 \) classifier depends on a \( c_2 \) classifier if \( c_1 \) object does not own a \( c_2 \) object/reference, the \( c_1 \) object is not associated with a \( c_2 \) object reference, and \( c_2 \) is the return type of a \( c_1 \) method or \( c_1 \) has created the \( c_2 \) object.

\[
\text{dep}(c_1, c_2) := \neg \text{ownedObj}(c_1, c_2) \land \neg \text{ownedRef}(c_1, c_2) \land \neg \text{assocRef}(c_1, c_2) \land (c_2 \in c_1.\{u_i\} \lor c_2 = c_1.m_j.ret)
\]

Overall example. Applying our definitions on Prog. 1, we took as output the UML class diagram of Fig. 2. On the contrary, the diagram generated by the professional ObjectAid UML recovery tool is presented in Fig. 1. Comparing the two diagrams, we observe that the diagrams differ in five out of the seven UML relationship arrows.
Table 1. The dataset that we used for the effectiveness evaluation of the UML Recoverer.

<table>
<thead>
<tr>
<th>ID</th>
<th>Num. of Classifiers</th>
<th>Num. of Fields</th>
<th>Num. of Method Arguments</th>
<th>Num. of Method Return-Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Concrete</td>
<td>Abstract</td>
<td>Interface</td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>23</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>23</td>
<td>13</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td>29</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>31</td>
<td>21</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>30</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>34</td>
<td>23</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>44</td>
<td>35</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Figure 1. The diagram recovered by ObjectAid for Prog. 1.

Figure 2. The diagram recovered based on our definitions for Prog. 1.

5 Experimental Evaluation

We implemented in Java the UML Recoverer research-prototype of our approach. We evaluate the effectiveness of the UML Recoverer on anonymized student projects against ground-truth UML diagrams and diagrams recovered by existing professional UML class recovery tools. The number of the classifiers of the projects ranges from 15 to 44 (Table 1) and the number of their UML relationships ranges from 20 to 170 relationships. Searching for existing (free to use) UML class recovery Eclipse plug-ins in the Eclipse Marketplace, we found that the most widely used tools currently are the ObjectAid\(^7\) and the UML Lab\(^8\). To assess the effectiveness of the recovered binary UML relationships, we compare them against manually extracted relationships via using the precision and recall metrics [16].

The precision results are depicted in the first chart of Fig. 3. We observe the precision of the UML Recoverer steadily equals 1.0 in all projects (independently of the project cases). On the contrary, the precision of the other tools ranges from 0.37 to 0.86 and from 0.04 to 0.53, respectively. The recall results are depicted in the second chart of Fig. 3. We observe the recall of the UML Recoverer ranges from 0.79 to 1.0. In particular, the lower the number of the abstract classes and the interfaces a project includes, the higher the recall of the UML Recoverer is. This is due to the fact that the UML Recoverer does not capture association relationships to late-binding cases. The recall of the other tools ranges from 0.43 to 0.85 and from 0.05 to 0.53, respectively.

To explain why the precision and the recall values of the two tools is very low in some cases, we inspected the numbers of the UML relationships recovered by the tools and we made the following observations. The two tools do not recover the aggregation and the composition relationships at all. In particular, the ObjectAid considers as dependencies/associations the relationships that are aggregations or compositions. The UML Lab considers as associations the relationships that are dependencies, aggregations or compositions. The number of the associations recovered by the UML Recoverer is slightly lower than the ground-truth number. The reason is the late binding to objects. In particular, there are methods in the student projects that accept as input objects of abstract classes/interfaces and the UML Recoverer identifies the association to abstract classes/interfaces but not to concrete classes.

6 Conclusion and Future Work

We formally defined a set of mappings between UML relationships and OO relationships via using static-code analysis exclusively. A future direction of our work is the comparison of our algorithm against UML class recovery approaches that apply dynamic-code analysis. Another inter-

---

\(^7\)https://www.objectaid.com/class-diagram

\(^8\)https://www.uml-lab.com/en/uml-lab/videos/reverse-engineering
Figure 3. The precision and the recall results for the three recovery tools.

Testing future direction would be the recovery of N-ary associations. Finally, the employment of semantic code analysis could further enrich the effectiveness of our approach.

References


Extracting Prerequisite Relations among Concepts from the Introduction of Online Courses

Zesong Wang
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
201822111920089@stu.hubu.edu.cn

Kui Xiao*
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
xiaokui1024@hotmail.com
(*Corresponding author)

Zeqing Qin
School of Computer Science
Hubei University of Technology
Wuhan, China
qzqhbut@163.com

Shihui Wang
School of Computer Science and Information Engineering
Hubei University
Wuhan, China
wsh@hubu.edu.cn

Abstract—Affected by the COVID-19 pandemic, teaching tasks have gradually shifted from offline to online, which expanded online education resources unprecedentedly. “Concept” is a professional vocabulary in the curriculum. Exploring the prerequisite relations among concepts is of great significance to educational planning. This research extracts concepts from online course introduction and proposes a mixed method for extracting concept prerequisite relations. Experiments on public data set show that this method outperforms existing ones. Tests were also carried out on the datasets of eleven schools, which proves that this model has good scalability.

Keywords- prerequisite relations; educational planning; online education resources

I. INTRODUCTION

Concepts are professional vocabulary covered in the course. Usually, dependencies among concepts exist. This is called prerequisite relations. In recent years, the extracting of prerequisite relations among concepts has become a focus of researchers. Prerequisite relations among concepts have played a significant role in many applied fields of smart education, such as curriculum planning and design [1,2], student knowledge status tracking [3], concept map building [4,5], learner ranking [6,7,8], document reading list generation [9,10], and so on.

With the rise of online education platforms such as MOOC, many universities launched their own courses on it, making online education resources richer in recent years. To make it easier for students choosing courses, online courses are generally equipped with a course introduction, which highlights the key knowledge of the course by condensing its core content. Based on concepts extracted from the online course introduction, a dependency extracting research was carried out. This research proposes a mixed method for extracting prerequisite relations among concepts. By analyzing the course introduction, the attributes of Wikipedia articles in accordance with corresponding concept, 10 different features are built and used to analyze whether prerequisite relations exist.

The structure of this article is as follows: Section 2 reviews the related work of prerequisite relation mining; Section 3 introduces the method of this article and constructs 10 different features. Section 4 conducts an experimental exploration of the proposed method. Finally, a conclusion of this research is drawn in Section 5.

II. RELATED WORK

Talukdar et al. [11] first study prerequisite relations mining between Wikipedia concepts. The author believes that if the Wikipedia article of concept B contains a link to concept A, then A may contain some background knowledge needed to learn before view B, which means A is a prerequisite of B. For these linked concept pairs, the author uses the MaxEnt classifier to predict the prerequisite relations among them.

Liang, C. et al. [12] propose a method based on concept reference distance (RefD) to predict the relations between two Wikipedia concepts. Specifically, each concept in Wikipedia can be replaced by its “set of related concepts”. If most of the concepts in the “set of related concepts” of concept B contain a link to concept A, and concept B is rarely cited by the “set of related concepts” of concept A. Then concept A may be a prerequisite of concept B. Zhou, Y et al. [13] use machine learning methods to predict the prerequisite relationship of Wikipedia concepts. The author establishes four sets of features of concept pairs, including link-based, category-based, text-based, and time-based features, six different classifiers are used for experiments. Sayyadiharikandeh et al. [14] propose a method for inferring the prerequisite relation between concepts based on Wikipedia clickstream data. Clickstream is the user’s operation log on the Wikipedia platform. This is the first time that researchers have used user interaction behavior to predict the prerequisite relation between concept pairs.

The above methods are all based on the content of the Wikipedia article. Besides, some researchers carry out research on the recognition of the curriculum concept prerequisite relations based on learning resources. Some analyzes the prerequisite relations between the curriculum concepts in the MOOC video [15]. Liang, C et al. [16] analyze the content of the university curriculum introduction to extract the main concepts and infer the prerequisite relationship between them, which is closely related to this research. However, the author only considers the influence of course attributes on prerequisite relations. In this research, course attributes and Wikipedia attributes are all adopted to identify the prerequisite relations between those concepts.
III. METHOD

A concept has many different attributes, such as the frequency and order in the online course introduction. In Wikipedia, each concept is an article with its own content. Links, clickstreams, classification, and other attributes of that article can be used for prediction. To better explore the prerequisite relations between these concepts, features are designed from the two aspects of “Course-based attributes” and “Wikipedia-based attributes”. In Fig.1, “graph algorithm” is a concept extracted from course introduction. Fig.1 shows the flow chart of our method.

A. Course-based attributes

In this part, features are designed by using the frequency, position of the concept in the online course introduction, and the learning order between courses. Four features are included in this part. The description of elements is defined as Table I.

<table>
<thead>
<tr>
<th>TABLE I. ELEMENTS RELATED TO COURSE-BASED ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
</tr>
<tr>
<td>C_i</td>
</tr>
<tr>
<td>w_i</td>
</tr>
<tr>
<td>Tif(C_i,w_i)</td>
</tr>
<tr>
<td>com(C_i)</td>
</tr>
<tr>
<td>exist(w_i)</td>
</tr>
<tr>
<td>order(w_i,w_j)</td>
</tr>
<tr>
<td>r(C_i,w_i)</td>
</tr>
<tr>
<td>Z(C_i,C_j)</td>
</tr>
</tbody>
</table>

- **Concept appearance**

The introduction of a course can be viewed as a series of “concepts”. exist(w_i) represents courses that have w_i in the introduction, Gw_i represents the probability that w_i appears in the course introduction where w_i appears.

\[
G_w_i = \frac{\text{exist}(w_i) \cap \text{exist}(w_i)}{\text{exist}(w_i)}
\]  

When introducing a new concept, the leading concept is introduced at the same time, which is also regarded as background knowledge. So when Gw_i is larger and Gw_j is smaller, w_i is more likely to depend on w_j. That is to say, the frequency of w_j in the course introduction that has w_i is very high. On the contrary, w_j as background knowledge, the frequency of w_i in the course introduction has w_j is not obvious. This also reasonably fits the general laws of cognition. Based on this assumption, the first feature is proposed to be “Gw_i - Gw_j”:

\[
C_w_i - G_w_j = G_w_i - G_w_j
\]  

- **Concept order**

The concepts contained in the course introduction can be regarded as an ordered list, and the position of each concept can be viewed as its index number.

The same concept may appear multiple times in a course introduction, the first appearance of the concept is taken as its position in this course introduction. order(w_i,w_j) represents the course whose position of w_i is before the position of w_j in the introduction.

\[
\text{order}(w_i,w_j) = \left| \text{exist}(w_i) \cap \text{exist}(w_j) \right|
\]  

In a course introduction, we believe that w_i is more likely to become the background knowledge of w_j, if the probability of w_i appearing before w_j is higher, and the probability of w_j appearing before w_i is smaller. The second feature is proposed to be “Cof(w_i,w_j)”:

\[
C_{of}(w_i,w_j) = C_{of}(w_i,w_j) - C_{of}(w_j,w_i)
\]  

- Concept in course

Each course has a corresponding introduction. The course C_i can be represented by a vector on the concept space \( \{w_1, w_2, \ldots, w_n\} \). The value in the vector is the tf-idf value of the different concepts in the C_i’s course introduction. E.g:

\[
C_i = \{0, 0.23, 0.014, 0, 0.56, 0, \ldots, 0.13, 0\}
\]

Assuming that w_i appears in the course introduction of C_i, w_i appears in the course introduction of C_j, and C_j needs to be studied before learning C_i, this course sequence possibly means that w_i depends on w_j. For two concepts, if they appear in multiple course pairs with a fixed sequence, then the relationship between these two concepts can be expressed as (5):

\[
C_r(w_i,w_j) = \sum_{i=1}^{n} \sum_{j=1}^{n} r(C_i,w_i)r(C_j,w_j)z(C_i,C_j)
\]  

r(C_i,C_j) indicates whether w_i is an important concept of C_j. When the tf-idf value of w_i in C_i is greater than a specified
threshold, it is an important concept of \( C_i \). In case of that, the value of \( r(C_i, w_j) \) is 1, otherwise the value is 0. For course \( C_i \), this threshold is defined as the average value of \( tf-idf\) of the concepts contained in \( C_i \)’s introduction.

\[
r(C_i, w_j) = \begin{cases} 1, & \text{if } Tid(C_i, w_j) > \frac{\sum_{w \in \text{con}(C_j)} Tid(C_i, w)}{\text{con}(C_j)} \\ 0, & \text{else} \end{cases}
\]  

(6)

\( Z(C_i, C_j) \) represents whether \( C_i \) depends on \( C_j \), and the value is 1 or 0, where 1 means that you need to study \( C_j \) before learning \( C_i \), and 0 indicates other cases. The third feature “\( Crf(w_j, w_a)\)” is defined as (7):

\[
Cr(w_j, w_a) = Cr(w_j, w_a) - Cr(w_j, w_u)
\]

(7)

- **Concept related to course**

Because the content of the online course introduction is often limited, some concepts that are closely related to the course cannot appear in the course introduction. For example, “knapsack problem” is the concept often explained in course “Algorithm Design and Analysis”, but the online course introduction may only include more coarse-grained concept like “dynamic programming method”. Therefore, we have to establish connections between the course and concepts that are not included in its introduction.

For a course \( C_i \) and a concept \( w_a \), \( w_a \) does not appear in the introduction of \( C_i \). However, \( w_a \) may have a strong connection with the concepts in the introduction of the course, their relevance can be expressed as (8):

\[
t(C_i, w_a) = \sum_{w_j \in \text{con}(C_i)} \frac{\text{exist}(w_a) \cap \text{exist}(w_j) \text{Tid}(C_i, w_j)}{\sum_{w_j \in \text{con}(C_i)} \text{Tid}(C_i, w_j)}
\]

(8)

\( w_j \) is a concept extracted from the course introduction of \( C_i \). The more frequent the appearance of \( w_a \) and \( w_j \) in same time, the higher the correlation between \( w_a \) and \( w_j \) is. Compare with all concepts included in \( C_i \), \( t(C_i, w_a) \) describes a relevance between the concept \( w_a \) and the course \( C_i \).

If two concepts \( w_a \) and \( w_b \) are in correspondence to such courses respectively, and there is a fixed order relationship between the course pair, then the relationship between these two concepts can be expressed as (9):

\[
C_s(w_a, w_b) = \sum_{j=1}^{n} \sum_{j=1}^{n} t(C_j, w_a) \ast t(C_j, w_b) \ast Z(C_j, C_j)
\]

(9)

On this basis, the fourth feature “\( Cs(w_a, w_b)\)” is defined as (10):

\[
Cs(w_a, w_b) = Cs(w_a, w_b) - Cs(w_a, w_u)
\]

(10)

- **Wikipedia-based attributes**

The attributes of concepts in Wikipedia are also used to identify the prerequisite relations between different concepts. Liang et al. [12] propose the idea “set of related concepts” for the first time and believe that for a pair of concepts, if there is a prerequisite relation between their related concept sets, it means that there is also a prerequisite relation between the two concepts.

We have innovated this approach. For \( w_a \), we regard the concepts that both belong to the same Wikipedia category (Category) as \( w_a \) and has a link relationship with \( w_a \) as the related concept sets of \( w_a \), denoted as \( S(w_a) \).

In what follows, we consider the prerequisite relation between concepts from the perspectives of “concept to concept”, “concepts to set” and “set to set” respectively. Some elements used in this section are defined as Table II.

<table>
<thead>
<tr>
<th>Elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S(w_a) )</td>
<td>Related concept sets of ( w_a )</td>
</tr>
<tr>
<td>( C_{sa} )</td>
<td>The set of concepts have clickstream point from ( w_a )</td>
</tr>
<tr>
<td>( C_{sa} )</td>
<td>The set of concepts have clickstream point to ( w_a )</td>
</tr>
</tbody>
</table>

- **Category information in Wikipedia**

In Wikipedia, each concept belongs to one or more categories. If the level of one category is above the level of another, the higher-level category usually contains more abstract concepts while the lower-level category usually contains concepts that are more concrete. These concrete concepts often rely on abstract concepts [1,13]. Therefore, we design the following features.

1) **Concept to Concept**

\( \text{len(root, } w_a) \) represents the root node in the Wikipedia category. \( \text{len(root, } w_a) \) represents the shortest path length from \( w_a \) to root node, which is also the level of the concept in the Wikipedia system. As is shown in Fig.2, \( \text{len(root, } w_a) = 2 \), \( \text{len(root, } w_b) = 2 \), \( \text{len(root, } w_c) = 3 \).

![Figure 2. Level of concepts in Wikipedia classification](image)

The larger the value of \( \text{len(root, } w_a) \), the more concrete the value of \( w_a \) is; The smaller the value of \( \text{len(root, } w_c) \), the more abstract the content of \( w_c \) is. If \( \text{len(root, } w_c) \) is smaller than \( \text{len(root, } w_b) \), it means that \( w_b \) has a higher level than \( w_a \) and its
content is more abstract than \(w_a\). Then \(w_b\) may be a prerequisite of \(w_a\), so we define the fifth feature “\(Waf(w_a,w_b)\)” as (11):

\[
Waf(w_a,w_b) = \text{len}(\text{root}(w_a)) - \text{len}(\text{root}(w_b)) \quad (11)
\]

2) Concept to Set

The average level between related concept set and concept are explored by using “the set of related concepts”. If the level of the set of \(w_a\)’s related concepts is below \(w_a\) on average, and the level of the set of \(w_b\)’s related concepts is above \(w_a\) on average, then we think that \(w_b\) is more likely to be a prerequisite of \(w_a\). And the sixth feature “\(Waf(w_a,w_b)\)” is defined as (12):

\[
Waf^*(w_a,w_b) = \frac{\sum_{w \in S(w_a)} \text{len}(\text{root}(w))}{|S(w_a)|} - \frac{\sum_{w \in S(w_b)} \text{len}(\text{root}(w))}{|S(w_b)|} \quad (12)
\]

3) Set to Set

We also design features between sets. We consider that for a pair of concepts \((w_a,w_b)\), if the average level of \(w_a\)’s related concept set is below the average level of \(w_b\)’s related concept set, then \(w_b\) is more likely to be a prerequisite of \(w_a\). So we define the seventh feature “\(Waf^*(w_a,w_b)\)” as (13)

\[
Waf^*(w_a,w_b) = \frac{\sum_{w \in S(w_a)} \text{len}(\text{root}(w))}{|S(w_a)|} - \frac{\sum_{w \in S(w_b)} \text{len}(\text{root}(w))}{|S(w_b)|} \quad (13)
\]

- **Clickstream in Wikipedia**

Wikipedia usually publishes user clickstream data logs in the past 30 months\(^1\). Clickstream refers to the action that a user browses a Wikipedia article immediately after another article. User usually clicks on the link of one article to jump to another article to continue brows in, those two articles are often closely related \([14]\). Clickstream records data of this type.

1) Concept to Concept

After browsing a Wikipedia article for a concept, people often continue to browse related concepts to view the background knowledge. If there is a clickstream from \(w_a\) to \(w_b\), \(w_b\) may be a prerequisite of \(w_a\). Therefore, we define the eighth feature “\(Wkf(w_a,w_b)\)” to identify the prerequisite relations between concepts.

\[
Wkf(w_a,w_b) = \begin{cases} 
1, & w_b \in C(w_a) \\
0, & \text{else} 
\end{cases} \quad (14)
\]

2) Concept to Set

Liang et al. \([12]\) use the link in the Wikipedia article of the concept to identify the dependency between concepts. We improve this approach with more precise clickstreams. Clickstream data is different from links, which are made by real users. Since people tend to browse the background knowledge when browsing new knowledge, we believe that if most of the \(w_a\)’s related concepts having a clickstream pointing to \(w_a\), but \(w_b\) is the opposite, then \(w_b\) maybe a background knowledge of \(w_a\). \(w_b\) is more likely to be a prerequisite of \(w_a\). Therefore, we define the ninth feature “\(Wkf^*(w_a,w_b)\)” as follows:

\[
Wkf^*(w_a,w_b) = \frac{\sum_{w \in S(w_a)} Wkf(w_a,w_b)}{|S(w_a)|} - \frac{\sum_{w \in S(w_b)} Wkf(w_a,w_b)}{|S(w_b)|} \quad (15)
\]

3) Set to Set

From the perspective of set to set relations, we use \(Out(w_a,w_b)\) to indicate the intersection of all clickstream from \(S(w_a)\) and \(S(w_b)\); \(In(w_a,w_b)\) represents the intersection of all clickstream to \(S(w_a)\) and \(S(w_b)\).

\[
Out(w_a,w_b) = \left( \bigcup_{w \in S(w_a)} C(w) \right) \cap S(w_b) \quad (16)
\]

\[
In(w_a,w_b) = \left( \bigcup_{w \in S(w_b)} C(w) \right) \cap S(w_a) \quad (17)
\]

If \(Out(w_a,w_b)\) is larger and \(In(w_a,w_b)\) is smaller, it means that users often browse related concepts of \(w_b\) after browsing related concepts of \(w_a\), but rarely continue to browse related concepts of \(w_b\) after browsing the related concepts of \(w_a\), which shows that \(w_b\) may be a prerequisite of \(w_a\). So we define the tenth feature as “\(Wkf^*(w_a,w_b)\)”.

\[
Wkf^*(w_a,w_b) = Out(w_a,w_b) - In(w_a,w_b) \quad (18)
\]

IV. EXPERIMENT

A. Datasets

Liang et al. \([16]\) crawled the data of 654 computer science courses from the online course websites of 11 Well-known universities, which include the introduction of each course and the learning order between courses\(^2\). Among these courses, 861 pairs of courses have fixed learning sequence. We will verify the proposed method on this data set. The data set were cleaned, we obtain 1312 concept pairs with dependencies and 2448 concept pairs without dependencies. Table 3 shows the detailed information of this set, “Concept prerequisite relations” represents the number of concept pairs that have dependencies.

<table>
<thead>
<tr>
<th>Universities</th>
<th>#Courses</th>
<th>#Course pairs</th>
<th>#Concept prerequisite relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caltech</td>
<td>41</td>
<td>56</td>
<td>461</td>
</tr>
<tr>
<td>Illinois</td>
<td>72</td>
<td>97</td>
<td>554</td>
</tr>
<tr>
<td>CMU</td>
<td>65</td>
<td>78</td>
<td>618</td>
</tr>
<tr>
<td>Iowa</td>
<td>38</td>
<td>50</td>
<td>395</td>
</tr>
<tr>
<td>Maryland</td>
<td>34</td>
<td>54</td>
<td>455</td>
</tr>
</tbody>
</table>

B. Evaluation Results

This research uses five cross-validations to evaluate the proposed method. Six commonly used machine learning classifiers were used to predict the prerequisite relations among concepts. They are Random Forest (RF), Naive Bayes (NB), Multilayer Perceptron (MLP), and Support Vector Machine (SVM), Logistic Regression (LR), and AdaBoost. All classifiers are implemented using python program and sklearn library, and the parameters are default ones. Specific experimental results are shown in Table 4.

It can be seen from Table 4 that the prediction results of different classifiers are quite different. The random forest classifier has the best performance. It is better than other classifiers in metrics such as Accuracy, Precision, Recall, and F1, reaching 84.18%, 80.66%, 73.18%, and 76.26%, respectively, which is similar to the conclusions of related studies [13,15].

Support vector machines (SVM) performs poorly, with index values such as Recall and F1 being only 15.13% and 24.16%. It is estimated that because the feature values are all specific values, and the range of these values is quite different, it is difficult to form a better hyperplane in the two types of samples to classify the samples. We will use random forests for following experiments.

C. Comparison With Baselines

We select three baseline methods for comparison. The first is the method of calculating the concept reference distance (RefD) proposed in [12]. The author used two ways to define the weight of each related concept. One is equal (the weight of all related concepts is 1), the other is tf-idf (the weight of all related concepts is their tf-idf value). We name them “RefD-equal” and “RefD-tfidf”.

The second method is a concept dependency recognition method based on optimization technology (CPR) proposed in [16]. This is the first time that the course learning sequence is used to calculate the prerequisite relations of concepts. The third method is the concept prerequisite relations prediction method (EPR) proposed in [13] using features such as link, category, text, and creation time. The specific experimental results are shown in Table 5.

<table>
<thead>
<tr>
<th>MIT</th>
<th>165</th>
<th>220</th>
<th>712</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSU</td>
<td>33</td>
<td>59</td>
<td>390</td>
</tr>
<tr>
<td>Princeton</td>
<td>16</td>
<td>20</td>
<td>292</td>
</tr>
<tr>
<td>PSU</td>
<td>77</td>
<td>98</td>
<td>479</td>
</tr>
<tr>
<td>Purdue</td>
<td>20</td>
<td>16</td>
<td>282</td>
</tr>
<tr>
<td>Stanford</td>
<td>93</td>
<td>113</td>
<td>711</td>
</tr>
</tbody>
</table>

D. Analysis of Feature Contribution

To explore the importance of each feature in the classification task, we analyze the contribution of them. Table 6 shows the changes in the average accuracy after removing each feature in turn. The contribution of “Wikipedia-based Attributes” is greater than that of “Course-based Attributes”. When the features based on Wikipedia were removed, the average accuracy falls by 9.27%, and when the features based on course attributes were removed, the average accuracy falls by 5.42%. This may be due to the number of features in “Based on Wikipedia” is slightly more than that in “Based on course”.

Among the features of “Course-based Attributes”, $Caf(w_s, w_t)$ contributed the most, and the average accuracy falls by 4.12% after removal. The contribution of $Crf(w_s, w_t)$ is the smallest, which may be due to that only few concept pairs were involved.

Among the features of “Wikipedia-based Attributes”, $Caf(w_s, w_t)$, $Cof(w_s, w_t)$, and $Csf(w_s, w_t)$ contributed the most, and the average accuracy falls by 5.42%, 4.41%, and 4.41% after removal, respectively. The contribution of $Waf(w_s, w_t)$, $Wsf(w_s, w_t)$, and $Wkf(w_s, w_t)$ is the smallest, which may be due to that only few concept pairs were involved.

The proposed method outperforms other methods in all metrics. It is concluded that neither RefD nor CPR uses machine learning classifiers in the classification task. The EPR method uses conventional classifiers to classify as in this article, and their performances are significantly better than RefD and CPR methods, which shows that artificially established features can indeed provide effective help for the recognition of dependencies between concepts.
E. Cross-School Testing

The overall data set is composed of data from 11 schools. To figure out the results of the intersection of different school data sets and the average accuracy under cross-school conditions. We first use the data sets of 11 schools to train the classifiers to explore the average accuracy of each school respectively. And a “cross-school test” is conducted to explore the scalability and adaptability of the model in the cross-school situation.

Fig.3 shows the experimental results. Take Caltech as an example, “In-school Training” represents the use of Caltech’s data set to train the classifier and test the accuracy of the school’s internal prediction; “Out-of-school Training” means using the data sets of other ten schools as the training set, and Caltech’s data set as the testing set to verify the accuracy of prediction.

![Figure 3. Cross-school testing](image)

It can be seen from Fig.3 that compared with the accuracy of training the classifier on the overall data set (84.18%), the accuracy of training the classifier on a single school data set of “In-school Training” is generally low. This is also because the size of the data set decreases when trainings were performed in single school.

The accuracy rate of each school in the “Out-of-school Training” is training is slightly lower than that of the “In-school Training”, but the gap is not very significant. This may be caused by the unbalanced division of the data set. On average, the accuracy of the “Out-of-school Training” to “In-school Training” ratio in 11 schools is 94.34%, close to 95%. This proves that our model has excellent scalability and universality. If the data volume is larger, the trained model can better adapt to the prediction of the prerequisite relation in an unknown situation.

V. CONCLUSION AND FUTURE WORK

This research proposes a new method to extracting prerequisite relations among concepts from online course introduction, using the course attributes and Wikipedia attribute design features together. Experiments show that this method is superior to existing baselines.

Due to the limitation of the data set, this research only conduct experiment in the field of computer science. In the future, we will create concept pair data sets from online courses in different majors and languages, further verify and improve the model we proposed. Besides, we will also try to analyze various types of learning resources such as online video subtitle data and textbooks, using them to extract the prerequisite relationships between concepts.

ACKNOWLEDGMENT

This research was supported by The National Natural Science Foundation of China (No.61977021), The Technology Innovation Special Program of Hubei Province (No.2018-ACA139,No.2019ACA144), The Research Project of Hubei Provincial Department of Education (No.D20191002).

REFERENCES

A Technical Capability Evaluation Model Based Concept and Prerequisite Relation in Computer Education

Jiwen Luo, Tao Wang, Junsheng Chang, and Xiaoting Guo
National Key Laboratory for Parallel and Distributed Processing
College of Computer, National University of Defense Technology
Changsha, 410073, China
Email: {luojiwen18, taowang2005, cjs7908, guoxiaoting18}@nudt.edu.cn

Abstract—Effectively assessing the results of users’ online learning and enhancing social recognition has become a major development direction for online education platforms. For computer education, this article constructs a technical capability assessment model. This model integrates professional concepts in the field of computer science and extracts knowledge concepts from educational resources. The model first extracts candidate concepts, then uses a graph propagation algorithm to quantify candidate concepts and obtains concepts from them, and finally uses prerequisite relationships to further quantify the concepts mastered by students. The model is tested in the real learning environment of 250 students. This model has been proved to own certain practicability and reliability by Kendall rank correlation coefficient, which is used as an evaluation index.

Index Terms—concept extraction; technical capability evaluation model; online learning

I. INTRODUCTION

In recent years, massive online open courses (MOOCs) have been developing rapidly, providing convenient education for more than 100 million users worldwide [1]. A survey from Coursera shows that MOOCs are indeed beneficial to learners who complete the course. As the survey implies, 61% and 72% of respondents reflected that MOOCs benefit education and career respectively [2]. However, with the rapid development of MOOCs, related problems and challenges have emerged as well, such as poor continuous learning, high dropout rate, lack of personalized training, insufficient practical training, and so on. Meanwhile, MOOCs platforms have been criticized for their low completion rate [3]. The average course completion rate of edX is only 5%, and the completion rate of China’s MOOCs platform—XuetangX is 4.5% [4].

To enhance the autonomy and continuity of users’ learning, MOOCs platforms have made some efforts to expand their social recognition. The MOOCs platform launched a micro-degree certification model. For example, Udacity’s data analysis micro-degree certificate is completed in cooperation with Facebook and MongoDB.

Besides, online education platforms and job search websites have started cooperation. In 2015, LinkedIn, an American professional networking site, announced the acquisition of Lynda, an online teaching site founded in 1995. LinkedIn’s CEO Jeff Weiner said, “The mission of LinkedIn and the mission of lynda.com are highly aligned. Both companies seek to help professionals be better at what they do. lynda.com’s extensive library of premium video content helps empower people to develop the skills needed to accelerate their careers.” When integrated with the hundreds of millions of members and millions of jobs on LinkedIn, lynda.com can change the way in which people connect to opportunity [5].

With the rapid development of information technology, computer professionals are often required to possess a variety of knowledge. Paying attention to the recruitment needs of computer professionals, we find that the requirements in the recruitment notice mainly focus on technical capabilities, including mastery of programming languages, technical frameworks, related tools, related project experience, and so on. For example, different programming languages, algorithms, data structures, computer networks, technical frameworks, front-end and back-end development, etc.

For online education platforms, providing competency certification and increasing employment opportunities will be the development direction of the platform to enhance its user stickiness. For users, obtaining certification on the education platform and increasing employment opportunities will be their motivation for continuous learning.

Therefore, to increase the user stickiness of online education platforms and promote better adaptive learning of students, this article proposes a technical capability evaluation model to measure the breadth of knowledge acquired by students for computer education. This model mainly evaluates the knowledge points that students have mastered on the online learning platform. For computer-related majors, skill points are a key factor in their employment.
The main contributions of this article are summarized as follows:

- Extract candidate concepts from online education resources.
- Evaluate the quality of candidate concepts and extract high-quality phrases to complete the concept extraction of educational resources.
- Mine the prerequisite relationships of concepts from an existing open-source knowledge base.
- Integrate the relationship between knowledge to quantify the score of each concept.
- Construct a technical ability evaluation model based on the above content.

The main innovations of this article are listed below:

- Integrate MOOCCube\(^1\) (A Large-scale Data Repository for NLP Applications in MOOCs) datasets and use graph propagation algorithms to label teaching resources in the computer education field.
- Combine the prerequisite relationship to quantify the score of concepts.
- Train a technical ability evaluation model based on the users’ online learning data in a real environment.

The mark of educational resources completed in this article can be used for follow-up tracking of students’ learning status, provide help to understand the mastery of specific knowledge points and serve as the basis for recommending courses and other recommended applications. On the one hand, the technical ability assessment model proposed in this paper can help users understand their knowledge. On the other hand, it can accurately recommend talents for the job market.

II. RELATED WORK

Related work mainly includes the following three aspects: student ability assessment in online education, curriculum concept extraction, and prerequisite relationship extraction.

**Capability Assessment.** In recent years, students’ ability models of online education mostly measure the change of students’ ability in the field of knowledge tracking [6]–[8], which evaluates the knowledge mastered by students based on the marked knowledge points. Some researchers excavate and analyze students’ learning activities [4] and make summative evaluations of students’ performance [9], [10]. There is also some corresponding work in the comprehensive evaluation of students that combines online learning process data and results data [11]. However, the evaluation of students’ technical ability in the field of computer education remains unsolved.

**Concept extraction.** About concept extraction in the field of natural language processing, researches mainly comprise supervised and unsupervised methods. Supervised methods are used to train classifiers. Unsupervised methods are commonly used in TF-IDF, TextRank, and so on. Different from keyword extraction, concept extraction of MOOCs resources is faced with fewer relevant documents, short texts, and fewer words (usually only the corresponding document introduction provided by teachers, which can be regarded as PPT with subtitles), especially the courses of computer major, which have the characteristics of domain knowledge extraction and are highly professional. There are few studies on keyword extraction of MOOCs resources, such as the course concept extraction [12] and course concept expansion [13].

**Prerequisite relationship extraction.** Inferring concept prerequisites from course dependencies or video-based course data are relatively new areas. Some of classical methods include CGL [14], CPR-Recover [15], MOOC-RF [16], and PREREQ [17].

To the best of our knowledge, our paper is the first to quantify concepts score based on prerequisite relationships after the completion of concept extraction, and apply it to student ability assessment.

III. OUR APPROACH

Based on the concept extraction of educational resources, we constructed a technical ability evaluation model, which includes a candidate concept extraction module, a prerequisite relationship extraction module, a candidate concept score quantification module, and a parameter fitting module. This is shown in Figure 1.

![Fig. 1. Overview of our approach.](#)

The candidate word extraction module mainly performs keyword extraction. The prerequisite relation extraction module extracts knowledge concepts with prerequisite relations from the existing knowledge base. The candidate concept scoring and quantification module use graph propagation algorithms to combine sequential knowledge concepts after constructing the knowledge graph. Perform quantitative scoring, and the parameter fitting module is result-oriented to test the impact of different functions and coefficient settings on the technical capability evaluation model.

The technical ability assessment model is designed to measure the technical ability of students. Technical ability is represented by the knowledge points mastered by students. Knowledge points are concepts acquired from educational resources. The acquisition of the concept mainly includes the

\(^1\)http://moocdata.cn/data/MOOCCube
following three steps. The first step is to extract keywords or candidate concepts from educational resources. The second step is to obtain high-quality phrases from candidate concepts. The third step is to evaluate high-quality phrases to obtain concepts. The third step depends on the human judgment after high-quality phrases were scored.

A. Candidate Concept Extraction

For computer-related majors, skill points are a key factor in their employment. The recruitment needs of Internet companies often lead to technical requirements. Therefore, we mine the skills that students may master from the courses they have completed.

We want to extract domain-specific concepts from these educational resources (such as lectures, video captions, knowledge introductions, etc.). First, we extract candidate concepts (i.e. keywords) from educational resources.

A course corpus $D$ is composed of $|D|$ courses in the same subject area.

$$D = \{\text{Course}_j\}_{j=1}^{|D|}$$

Each Course is composed of $m_j$ educational resources.

$$\text{Course}_j = \{\text{M}_{ij}\}_{i=1}^{m_j}$$

Course concept $C$ can be considered as a collection of topics taught in the course. Formally, each concept in the set of course concepts $C$ can be expressed as a candidate, which is defined as a k-gram in $D$.

The problem could be formally defined as: given a set of educational data in one domain, extract domain-specific concepts $C$ from $D$. $C$ is a collection of concepts.

$$C = \{\text{candidate}_i\}_{i=1}^{|C|}$$

The pseudocode for candidate word extraction (i.e. extract $C$ from $D$) is Algorithm 1. We employ the linguistic pattern $((A ~ | ~ N)+ | (A ~ | ~ N) */(NP)?(A ~ | ~ N)*N)$ [18] to determine whether a candidate word is a noun. The isNoun() function is used in pseudocode. Candidate concept extraction mainly includes preprocessing, word segmentation, part-of-speech tagging, judging whether it is in the vocabulary and whether it satisfies the defined linguistic pattern.

Through the above algorithm, we complete the candidate word extraction, but not all phrases extracted are domain-specific. For example, basic theory is a good phrase, but it is not a domain-specific concept. A domain-specific concept should satisfy (1) Phrasal: it is a semantically and syntactically correct phrase. (2) Informational: it is a scientific or technical concept related to the course in $D$ [12]. Therefore, we construct a weighted undirected graph and use the graph-based propagation algorithm to sort the vertices of the graph to identify high-quality candidate words.

Besides, not all phrases from candidate concepts extraction are domain-specific, e.g., the basic theory is a good phrase, but it is not a domain-specific concept. A high-quality phrase should combine the phraseness and informativeness information [19].

Algorithm 1 Candidate concept extraction

**Require:** course corpus $D$

**Ensure:** course concept $C$

1. preprocess educational material $M$. remove special characters and convert it to lowercase

   $$M = \text{process}(M)$$

2. for $m$ in $D$ do

   3. $\text{tmp} = \text{cut}(M)$

   4. $\text{seg} = \{(t.\text{word}, t.\text{flag}) \text{ for } t \text{ in } \text{tmp}\}$

   5. $n = \text{len}(\text{seg})$

   6. for $i$ in range($n$) do

      7. $\text{phrase} = \text{seg}[i][0]$

      8. $\text{flag} += \text{seg}[i][1]$

      9. for $j$ in range($i + 1, \text{min}(n + 1, i + 7)$) do

         10. if $\text{phrase}$ not in $\text{res}$ and $\text{phrase}$ in $\text{vocabs}$ and $\text{isNoun}(\text{config}, \text{flag})$ then

            11. $\text{candidate}.\text{add}(\text{phrase})$

         end if

      end for

   end for

end for

end for

19: end for

20: return candidate

We construct a weighted undirected graph and use the graph-based propagation algorithm to sort the vertices of the graph to identify high-quality candidate words (i.e. concepts or key points) [12].

B. Construction of Graph

First, a weighted undirected graph $G = (V, E)$ was constructed, where $V$ is the vertex set of $G$ and $E$ is the edge set of $G$. Each vertex $V$ is a phrase $p_i \in P$ with a quality score $Q(p_i)$. $P$ is a set of high-quality phrases extracted from $D$, and $P = \{p_i\}_{i=1}^{|P|}$

$$Q(p_i) = \begin{cases} 1, & \text{if } p_i \text{ in concept seed} \\ 0, & \text{if } p_i \text{ not in concept seed} \end{cases}$$

Concept seed is a high-quality professional vocabulary based on the text material to be processed. For each edge $(p_i, p_j) \in E$, its weight $w(p_i, p_j)$ is the semantic relatedness of the two phrases $p_i$ and $p_j$. We used a pre-trained BERT [20] model to get pre-trained word contextual embeddings, and then obtained the semantic representation of each phrase via the vector addition of its word vectors. Finally, the semantic relatedness of two phrases is defined as the cosine similarity of their vectors.

C. Graph Propagation Process

The sorting method based on propagation was performed on a graph $G$. It is assumed that the high-confidence concept in the graph can propagate its confidence value to neighbor
nodes with high semantic relevance to discover other potential domain-specific concepts [21]. This section contains steps to obtain high-quality phrases from candidate concepts.

Each vertex \( p_i \) has a confidence score \( conf(p_i) \) of being a domain-specific concept and \( conf^k(p_i) \) is the score of \( p_i \) in the \( k-th \) iteration of the propagation.

We set the initial confidence score of each vertex as \( conf^0(p_i) = 1 \). The propagation functions are defined as:

\[
conf^{k+1}(p_i) = 1/Z \left( \sum_{p_j \in A(p_i)} s^k(p_j, p_i) \right)
\]

The voting score function as follows.

\[
s^k(p_j, p_i) = opf(p_i, p_j) \cdot Q(p_j) \cdot e(p_i, p_j) \cdot conf^k(p_j)
\]

where \( s^k(p_j, p_i) \) is the voting score that \( p_j \) propagates to \( p_i \) in the \( k-th \) iteration which is determined by \( opf(p_i, p_j) \), \( Q(p_j) \), \( e(p_i, p_j) \), and \( conf^k(p_j) \).

\( opf(p_i, p_j) \) is the overlapping penalty between \( p_i \) and \( p_j \). If \( p_i \) and \( p_j \) contain one or more identical words, we say they are overlapping and should be penalized during the propagation process.

\[
\begin{align*}
opf(p_i, p_j) &= 1, & \text{if not overlapping} \\
opf(p_i, p_j) &= \lambda, & \text{if overlapping } \lambda \in (0, 1)
\end{align*}
\]

\( Q(p_j) \) is the quality score of \( p_j \), \( e(p_i, p_j) \) is the semantic relatedness between \( p_i \) and \( p_j \), and \( conf^k(p_j) \) is the confidence score of \( p_j \) in the \( k-th \) iteration.

\( con^{k+1}_f(p_i) \) is the new confidence score of \( p_i \), which is dependent on the average voting score of vertexes in \( A(p_i) \). \( A(p_i) \) is the vertex set that will propagate the voting scores to \( p_i \) in each iteration. After each iteration, the confidence scores should be normalized, so \( Z \) is the normalization factor.

To determine when the iteration process stops, a termination set \( F \) was introduced. \( Ar^k(F) \) is the average ranking of concepts in \( F \) after the \( k-th \) iteration, if \( Ar^{k+1}(F) > Ar^k(F) \) then the propagation process terminates.

D. Prerequisite Relationship Extraction

To reasonably determine the weight of concepts extracted from educational resources, we need to consider the concept relationships. For example, if a user has mastered the width-first search algorithm, he or she may already have mastered adjacency lists before. In other words, mastering the concept of B means that a user has mastered the concept of A. That is to say, A is the prerequisite concept of B. Prerequisite relationship is also called pre-order relationship.

Prerequisite relationships are incorporated into the assessment model, which makes the ability scores depend on not only the number of learning courses but also the difficulty and quality of learning. We use files in the MOOCCube database to extract the prerequisite relationship in the computer science field based on a simple text matching algorithm.

E. Technical Capability Evaluation Model

The technical ability scoring model is defined as:

\[
Score = \sum_{i=1}^{n} \alpha \sum_{j=1}^{m} sc(concept_i)
\]

\( concept_i \) stands for high-quality conceptual phrase (i.e. concept). \( n \) represents the number of completed learning materials. \( m \) represents the number of concepts possessed by a learning material, and \( sc() \) is a function to quantify the concepts score.

1) The parameter \( \alpha \): \( \alpha \) can be set according to the difficulty of the learning materials. (1) According to the time of completing the learning material, the outlier data can be removed, and the average value can be calculated and then normalized. (2) If the learning material has a marked difficulty coefficient, \( \alpha \) can be obtained after quantifying the difficulty coefficient. The experiment in this article used the second method.

2) The function \( sc() \): Inspired by the graph propagation algorithm, concepts that are more related to other concepts should be more basic and common knowledge points. To explore the possible effects of different functions on the results, our experiment tested different functions for performance evaluation.

As for the setting of \( \alpha \) and \( sc() \), this article is result-oriented to make the constructed scoring model relatively reliable.

IV. EXPERIMENTS

A. Datasets

The experiments were based on the data of EduCoder\(^2\), an actual web-based online programming teaching platform.

\( \text{Fig. 2. A task challenge page belongs to EduCoder} \)

In concepts extraction, 9243 challenge tasks of the platform were used for candidate concepts extraction. The text description of challenge tasks was mainly used here. In MOOCCube, 358 pairs of prerequisite relations about computer science concepts were extracted. On the EduCoder platform, the

\(^2\text{https://www.educoder.net/}\)
technical capability evaluation model was trained based on the data of the 250 most active users, these users have completed 50190 tasks in total.

When extracting keywords, the input text was tokenized and annotated with part-of-speech (POS) tags by jieba\(^3\). BERT is a pre-trained model in the candidate concepts ranking module, and the seed file selected 4884 computer science concepts extracted on MOOCube.

In our experiments, a total of 548 candidate words were extracted, which were matched to the corresponding challenge task. After the candidate words were extracted, the graph propagation algorithm was used to calculate the confidence score of the candidate words, where the penalty factor \(\lambda\) was set to 0.5. We used the graph propagation algorithm in the concept sorting module to calculate the score of each concept, deleted concepts with a score of 0, and manually deleted some concepts that were not knowledge points. Finally, we got 412 concepts.

The 412 concepts were matched with 358 pairs of prerequisite concepts extracted in MOOCube, and 85 pairs of prerequisite concepts were identified in the concepts we extracted. Table I shows some prerequisite concepts in MOOCube by extraction.

<table>
<thead>
<tr>
<th>Prerequisite concept</th>
<th>Subsequent concept</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial search</td>
<td>breadthfirst search</td>
</tr>
<tr>
<td>array</td>
<td>hash function</td>
</tr>
<tr>
<td>search</td>
<td>insertion sort</td>
</tr>
<tr>
<td>computational science</td>
<td>dynamic allocation</td>
</tr>
<tr>
<td>realm name</td>
<td>wide area network</td>
</tr>
</tbody>
</table>

Before determining the parameter \(\alpha\) and the function \(sc()\), the score of a concept that is in the prerequisite relationship pairs will be recalculated. If A is the prerequisite concept of B, then the score of concept B is the sum of the scores of all concepts in set A, where A is a collection of concepts and B is a single concept.

### B. Evaluation Metrics

Kendall’s Tau [22] was used to compare the technical ability score ranking of users evaluated by this model and the actual user score ranking on the online education platform. Kendall’s Tau is a non-parametric measure of relationships between columns of ranked data.

The Tau correlation coefficient returns a value of \(-1\) to \(1\). \(-1\) means that the rank correlation of the two sequences is completely opposite, \(0\) means that the sequence is not related, and \(1\) means that the sequence is completely consistent.

\[
Kendall's \ Tau = \frac{(C - D)}{N} \quad (9)
\]

\(C\) is the number of concordant pairs and \(D\) is the number of discordant pairs. With \(n\) denoting the number of elements in the list, \(N\) is the total number of element pairs and can be calculated as below:

\[
N = \frac{1}{2n(n - 1)} \quad (10)
\]

### C. Experiment Results

\(\alpha\) was set to 0-10, which corresponded to the difficulty of challenge tasks. The difficulty of tasks in the EduCoder platform was divided into five levels, which can also be regarded as the difficulty levels of educational resources.

Different function types were assigned to \(sc()\). Table I shows the calculation results of a linear function and an inverse proportional function. The value in the Table II to control sequence is Kendall’s Tau value, the absolute value of the function coefficients does not affect on the result. The conclusion is consistent with Kendall’s Tau’s consideration of the relativity of ranking.

The paper lists four examples of positive and negative, and the coefficients of the two functions are taken as \([-1.0,1.0]\]. The data in Table II is calculated the 250 users when the prerequisite relationship between concepts were considered.

<table>
<thead>
<tr>
<th>Function</th>
<th>Linear function</th>
<th>Inverse proportional function</th>
</tr>
</thead>
<tbody>
<tr>
<td>(sc())</td>
<td>0.749</td>
<td>0.749</td>
</tr>
<tr>
<td>(sc())</td>
<td>0.485</td>
<td>0.485</td>
</tr>
</tbody>
</table>

Fixed the technical ability evaluation model \(Score\) (let \(sc()\) be a linear function, and the function coefficient is 1) and got the user data of the top 50, top 100, top 150, top 200, and top 250 in EduCoder. We separately calculated Kendall’s Tau value between the user ranking in the Score model and the actual user ranking without considering the prerequisite relationship. Table III shows that when the prerequisite relationship between concepts is considered, the score calculated by the score model is closer to the actual ranking.

<table>
<thead>
<tr>
<th>User number</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>No prerequisites relations considered</td>
<td>0.653</td>
<td>0.652</td>
<td>0.611</td>
<td>0.542</td>
<td>0.515</td>
</tr>
<tr>
<td>Prerequisites relations considered</td>
<td>0.647</td>
<td>0.654</td>
<td>0.614</td>
<td>0.552</td>
<td>0.521</td>
</tr>
</tbody>
</table>

We have counted the data of the user who ranked first on the EduCoder platform. According to the technical ability evaluation model, the top ten concepts obtained by the user after learning on the platform are: data, methods, objects,
arrays, graphs, strings, input and output, algorithm, model, and attribute.

D. Discussion

The effectiveness of the technology capability evaluation model mainly depends on the concepts extracted from educational resources.

This model integrates a large-scale MOOCs database and takes the prerequisite relationship between concepts into account. When Kendall’s Tau is greater than 0.5, it indicates that the sorted list has a certain similarity.

By comparing various simple functions $sc()$, it can be found that the score of each concept has a low impact on the score of the user’s final technical ability evaluation model. What really matters is the ranking of scores between concepts. If a better evaluation model of the users’ technical abilities was expected, the order of user learning sequences and knowledge mining need to be comprehensively considered. The sequence and relevance of concepts can be used to initialize the concept graph differently to improve the graph propagation algorithm in order to better build the technical capability evaluation model.

The model has certain limitations. Our experiment believed that when a user completes a task (or learning a course) in an online education platform, the user has mastered the knowledge concept corresponding to the task (or the course) and accumulates scores for the corresponding concept. It does not take into account the time and quality of the user’s completion, so the description of the user’s technical ability is more like a description of the technical breadth of the user. In addition, there’s still plenty of scope for improvement to distinguish whether the keywords extracted from the model belong to the knowledge concept of a specific domain.

V. CONCLUSION AND FUTURE WORK

To improve the social recognition of online education platforms, this paper constructed a technical ability evaluation model in the computer field based on the learning records of students on the online education platform. The technical ability evaluation model based on concepts and prerequisite relations. Its construction is mainly completed by extracting candidate concepts, calculating the importance of candidate words using graph propagation algorithm, recalculating the importance of candidate words using pre-order relations, and constructing a technical ability score model. The prerequisite relations in our research were extracted from the concept relations in the field of computer science and technology in MOOCCube. Therefore, the research conclusions of this paper are only applicable to the field of computer education. But these research ideas can be extended to other fields.

In the future, the model is planned to be deployed on the online education platform. For online education platforms, this model can serve as a basis to provide students with ability certification, and can also provide students with job-hunting advice. A more promising direction in the future is to combine with job-hunting websites to achieve the matching of vocational skills and employment recommendations.

ACKNOWLEDGMENT

Thanks Yue Zhou and Ruolin Sun for their effective suggestions during the writing process.

REFERENCES

Triangle Counting by Adaptively Resampling over Evolving Graph Streams

Wei Xuan†‡ and Huawei Cao† and Mingyu Yan† and Zhimin Tang† and Xiaochun Ye† and Dongrui Fan†
†Institute of Computing Technology, Chinese Academy of Sciences, China
‡University of Chinese Academy of Sciences, China
Email: {xuanwei, caohuawei, yanmingyu, tang, yexiaochun, fandr}@ict.ac.cn

Abstract—Triangle counting is a fundamental graph mining problem, widely used in many real-world application scenarios. Due to the large scale of graph streams and limited memory space, it is appropriate to achieve the estimation of global and local triangles by sampling. Existing streaming algorithms for triangle counting can be generalized into two categories. One is Reservoir-based methods employing a fixed memory budget, whose size is difficult to set for accurate estimation without any prior knowledge about graph streams. The other is Bernoulli-based methods, which sample edges by a given probability with uncontrollable memory budget. In this work, we propose a novel and bounded-sampling-ratio method, called BSR-Sample, by adaptively resizing memory budget upwards over evolving graph streams. BSR-Sample can keep the sampling ratio always greater than or equal to a specified threshold with available memory space. Then, we design BSR-TC, a single-pass streaming algorithm for both global and local triangle counting, based on BSR-Sample. Experimental results show that BSR-TC achieves accuracy of at least 99.8% for global triangles, when the ratio of initial memory budget to whole graph streams ≥ 0.002% and given threshold = 20%. And our proposed BSR-TC can gain more advantage than the state-of-the-art algorithms over the continuous growth of graph streams.

Index Terms—Evolving Graph Streams, Triangle Counting, Bounded Sampling Ratio

I. INTRODUCTION

There has been a burst interest in graph streams in recent decades, covering lots of real-world application scenarios, such as social networks, E-commerce, traffic networks [1], [2]. Graph streams are a continuous sequence of data items, often abstracted as edge streams, expressing entities and the relationships between these entities [3]. Triangle counting has many important applications and is widely used in graph data mining. For instance, the number of triangles can be used to detect communities [4], study clustering coefficient and connectivity [5] of social networks to improve user experience, discovery spam emails [6] to ensure the safety of customer services, etc.

Due to the large scale of graph streams and limited memory space, it is almost infeasible to calculate the real number of triangles by storing the entire graph streams. On the one hand, it will consume much more time on the communications between memory and secondary storage by storing the graph streams data into secondary memory. On the other hand, the fully dynamic characteristics of evolving graph streams also induces the difficulties of accurately counting global and local triangles within limited memory space. In the past few decades, there have been plenty of research [7]–[11] on stream sampling methods, which sample a small population of graph streams to compute the number of triangles as accurate as possible.

According to whether memory budget consumed by various sampling methods is fixed or not, we can classify the sampling methods into two categories: Reservoir-based sample and Bernoulli-based sample. The former will initially set a fixed memory budget, which stores the uniform sample chosen from graph streams. However, its sampling ratio monotonously decreases along the growing graph streams after the number of input edges exceeds memory budget. On the contrary, each member of the population has the same probability to be chosen and the inclusion variables are jointly independent in the Bernoulli-based sample. Thus, the memory budget used by Bernoulli-based sampling can vary in principle from 0 to the entire population size, which may exceed the available memory space and cannot be bounded to an expected value.

So far, existing streaming algorithms for triangle counting either fail to maintain a stable sampling ratio or controllable memory budget with the growth of graph streams. Taking full advantage of the characteristics of bounded sampling ratio and efficient utilization of memory budget, we propose a novel sampling method, called bounded-sampling-ratio sample (BSR-Sample), to maintain the sampling ratio greater than or equal to a specified threshold, when there is enough memory budget. The main contributions of this paper are as follows:

- Propose a novel and general sampling method, BSR-Sample, to keep the sampling-ratio great than or equal to a specified sampling ratio threshold, when available memory is enough large. To the best of our knowledge, BSR-Sample is the first attempt to adaptively increment memory budget with the continuous growth of graph streams. The highlight of this method is that we are able to maintain a bounded-sampling-ratio, without requiring any prior knowledge about the scale of graph streams.
- BSR-TC, a streaming algorithm for both global and local triangle counting over evolving graph streams, is proposed based on BSR-Sample. Compared with previous work for triangle counting, it is capable of discovering more triangles and attain higher accuracy by adaptively...
resizing memory budget upwards.

- Experimental results performed on real-world datasets show that BSR-TC can obtain more accurate estimation than the state-of-the-art sampling methods, with the growth of graph streams. Meantime, BSR-TC can keep stable results for both global and local triangle counting with the same specified threshold, regardless of differently initial memory budget.

The rest of this paper is organized as follows. In Section II, we review the related works. Then, we discuss the motivation of our work in Section III. In Section IV, BSR-Sample and BSR-TC are proposed and introduced in detail. In Section V, experiments are conducted using real-world datasets. In Section VI, we conclude our work.

II. RELATED WORK

In this section, we mainly introduce two categories of sampling methods, Bernoulli-based sample, and Reservoir-based sample, for triangle counting over graph streams. They are distinguished by whether consumed memory budget is fixed or not, as illustrated in Table I.

A. Reservoir-based sample

These sampling methods set a fixed memory budget, which is of importance considering the limited memory space. Vitter discussed optimized sampling algorithms in details based on the naive reservoir sampling method, and these optimizations improved the speed by an order of magnitude [12]. The main idea is to skip over a number of records rather than process all the records, reducing the called number of random number generators. However, the optimized Reservoir-based sampling methods are not suited for triangle counting, because they fail to update the estimations for every edge. Gemulla et al. further proposed a novel sampling method based on the naive reservoir sampling, called Random Pairing (RP), which handled both edge insertions and deletions for graph streams by the strategy of using future inserted edges to compensate for previous deletions [13]. [14] utilized temporal locality, where future edges were more likely to form triangles with recent edges than older ones, to improve the estimation accuracy. TRIEST was the first one to estimate triangles in fully-dynamic graph streams, involving both edge insertions and deletions by Reservoir-based sampling methods and its variants [15]. [16] proposed a family of algorithms for global and local triangle counting, called ThinkD, to further improve TRIEST by leveraging unsampled edges to update the estimations of triangles.

B. Bernoulli-based sample

The Bernoulli-based sampling method is relatively simple and efficient for it just needs an initial sampling probability to sample edges over graph streams. Therefore, this sampling method attracts considerable attention to count triangles in evolving graph streams. Ahmed et al. proposed a general sampling framework called graph sample and hold (gSH) for big-graph analytics by one single pass [17]. The gSH utilizes different sampling probabilities based on the graph properties of interest, e.g. gSH(p, q) samples the current arriving edge with probability p when it depends on previously sampled edges, otherwise holds the edge with probability q. Later Ahmed et al. proposed a new framework called graph priority sampling (GPS) for sequentially sampling over evolving graph streams [18]. Two estimation approaches are proposed to attain unbiased estimation of various graph properties, which are post-stream estimation and in-stream estimation. Lim et al. proposed a memory-efficient and accurate method for local triangle estimation over graph streams, called MASCOT [19]. It achieves best performance of both accuracy and memory efficiency of local triangle counting, by the means of “unconditional counting before sampling”.

III. MOTIVATION

The sampling ratio of Reservoir-based sample will monotonically decrease after the size of arriving edges exceeds the capacity of memory budget, which inevitably affect the estimation results, as shown by Figure 1. The memory budget of Bernoulli-based sample is not fixed or monotonically increases over the evolving datasets. Figure 2 shows that the sample size of Bernoulli-based sample fluctuates around the real value. Therefore, for the Bernoulli-based sample, it is difficult to allocate appropriate memory space for triangle counting over evolving graph streams.

Considering the characteristics of graph streams, we obtain observations which pose huge challenge for accurate triangle counting.

Observations:

- In real application scenarios, the scale of graph streams is unknown in advance.
- The evolving graph streams usually grow upwards as a whole.
- Memory space is limited to store all edges of graph streams.

These observations above lead to new challenges for global and local triangle counting over evolving graph streams, as described below.

Proposed problems:

- How to maintain an appropriate sampling ratio and how much memory space to allocate for accurate global and local triangle counting, without knowing any prior knowledge of evolving graph streams?

Goals:

- Maintain a bounded-sampling-ratio to achieve accurate and stable estimation of global and local triangle counting.

<table>
<thead>
<tr>
<th>Sampling Method</th>
<th>Sampling Ratio</th>
<th>Memory Budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSR-Sample (Proposed)</td>
<td>≥Threshold</td>
<td>Adaptive</td>
</tr>
<tr>
<td>Reservoir-based sample</td>
<td>Decreased</td>
<td>Increased</td>
</tr>
<tr>
<td>Bernoulli-based sample</td>
<td>Fixed</td>
<td>Fixed</td>
</tr>
</tbody>
</table>

TABLE I: Comparison of sampling methods. Note that the number of current edges is more than initial reservoir size and available memory space is enough large for the sake of simplicity.
Fig. 1: The blue line represents the number of discovered triangles, and the green line represents global error. For Reservoir-based sampling methods, both the number of discovered triangles and the accuracy of global triangles are positively correlated with memory budget.

![Graph showing correlation between memory budget and global error](image)

Fig. 2: The sample size of Bernoulli-based methods fluctuates around the real one with sampling probability $p=0.4$.

- Control Consumed memory budget to improve the efficiency of memory usage.
- Develop single-pass streaming algorithm to avoid multi-repeated and redundant operations.
- And no need to access slower secondary storage.

In general, Reservoir-based sample may induce the sampling ratio to monotonically decrease, which will damage the accuracy of estimations over evolving graph streams. Bernoulli-based sample cannot accurately calculate the size of memory budget. Therefore, we propose a novel Reservoir-based sample method, called BSR-Sample, which keeps a bounded sampling ratio and takes advantage of both methods above. BSR-Sample can maintain the sampling ratio greater than or equal to a given threshold and enable the memory budget to adaptively increase under control within available memory space. The comparison of the three sampling methods is illustrated in Table I.

IV. DESIGN AND ANALYSIS

In this section, we firstly introduce the overview of BSR-Sample and BSR-TC. Then, we show the implementation of these algorithms in detail.

A. Overview

Our proposed BSR-Sample always maintains the sampling ratio greater than or equal to a specified sampling-ratio threshold by leveraging multi-sets of Reservoir-based sample. Then, we propose BSR-TC using BSR-Sample method for triangle counting. As illustrated in Figure 3, BSR-Sample takes edges from graph streams as input, and outputs a set of sampled edges to BSR-TC for both global and local triangle counting.

- **Estimation.** BSR-Sample first estimates the number of global and local triangles for each arriving edge from the evolving graph streams, rather than samples them. Here, we call this mechanism as “first counting, then sampling”, which improves the estimation accuracy by leveraging more edges to participate in statistics analysis.

- **Sampling.** The whole sampling set of BSR-Sample is divided into two parts: the current sampling set and the sampling pool. The former is used to maintain dynamically updated edges, which are sampled based on the naive Reservoir-based sample. The sampling ratio of the current sampling set is always greater than the specified threshold. BSR-Sample will remove the sampled edges into sampling pool once it equals the threshold, and allocates new memory budget as the current sampling set.

- **Checking.** In this procedure, BSR-Sample determines whether to enable the current sampling set join into the sampling pool. When the sampling ratio of the current sampling set equals the specified sampling-ratio threshold, BSR-Sample will remove the entire current sampling set into sampling pool, and then create a new sampling set to substitute for current sampling set. Through adaptively incrementing the current sampling set, BSR-Sample is capable of keeping a bounded-sample-ratio when the available memory is large enough. And BSR-Sample can ensure the estimation accuracy without requiring any prior knowledge about evolving graph streams.

B. Algorithm Description

Here, we first introduce the sampling method of proposed BSR-Sample. Then, we analyze how to estimate global and local triangles by BSR-TC over evolving graph streams. We utilize the naive Reservoir-based sample for the current sampling set to sample edges.

Here, $\Theta$ is the totally available memory space. $S_c$ is the current sampling set, $S_p$ is the sampling pool, and $S_p(\text{th})$ is the $\text{th}$ single sampling set. $S = S_c \cap S_p$, $N^S_u$ is the set of neighbors of the node $u$ in $S$. $M$ is the initial memory budget. $R$ is a specified sampling-ratio threshold. Let $M/R$ is $T$. Note that, to simplify the description, we initialize the current sampling set and each single sampling set in the sampling pool to a same size.

- **CountTriangle (Lines 8-20 of Algorithm 2).** In this function unit, CountTriangle first checks whether each node of the arriving edge $(u, v)$ is contained in $S$ (lines 9-14),
Algorithm 1: Bounded-Sampling-Ratio Sample (BSR-Sample)

**Input:** (1) \(E(1), E(2), \ldots\): a graph stream;
(2) \(M\): initial memory budget;
(3) \(R\): specified sampling ratio threshold.

**Output:** \(S\): a set of sampled edges.

1. for each new arriving edge \(e_t = (u, v)\) do
2. \(SampleEdge((u, v), S_c)\).
3. \(CheckRatio((u, v), S_c, S_p)\).
4. end

Function \(SampleEdge(e_t, S_c)\)

1. \(t_c \leftarrow t \% M\)
2. if \(t_c \leq M\) then
3. \(S_c \leftarrow S_c + \{(u, v)\}\)
4. else if a generated random number \((0, 1) \leq M/t_c\) then
5. choose a random edge \((m, n)\) from \(S_c\)
6. \(S_c \leftarrow S_c \setminus \{(m, n)\}\)
7. \(S_c \leftarrow S_c + \{(u, v)\}\)
8. end
9. end
10. end

Function \(CheckRatio(e_t, S_c, S_p)\)

11. \(t_c \leftarrow t \% M\)
12. if \(M/t_c = R\) and \(\Theta \geq M\) then
13. remove the current sampling set \(S_c\) into sampling pool \(S_p\)
14. create a new \(S_c\) with size \(M\)
15. \(\Theta \leftarrow \Theta - M\)
16. end
17. end

End

which is the output of BSR-Sample in Algorithm 1. Then, we count the common neighbor \(N_{u,v}^{S_c}\) of nodes \(u\) and \(v\)(line 15). For each node \(c\) in \(N_{u,v}^{S_c}\), we update both global and local triangle counting by \(1/p_{cuv}\) (lines 16-19). Note that \(S_c\) and \(S_p^{(i)}\) in \(S_p\) are produced by the naive Reservoir-based sample, respectively. Therefore, BSR-TC is unbiased for triangle counting, where the expected value equals the real number of triangles. To compute the probability \(p_{cuv}\) that BSR-TC discovers the triangle \((c, u, v)\), we divide discovered triangles into 4 types, depending on the positions (\(S_c\) or \(S_p\)) of edges \((u, c)\) and \((v, c)\). When a new edge \(e_{t+1} = (u, v)\) arrives and forms a triangle with a node \(c\), \(p_{cuv}\) is calculated by following formula (1).

\[
p_{cuv} = \begin{cases} 
\min\{1, \frac{M}{t_c} \times \frac{M-1}{T-1}, \{(u, c), (v, c)\} \in S_c \\
\min\{1, \frac{M}{t_c} \times R, \{(u, c), (v, c)\} \in S_c \cup S_p^{(i)} \}
\end{cases}
\]

V. Experiments

We show that BSR-TC suffices to provide accurate estimation for both global and local triangle counting, without requiring any knowledge about graph streams. Compared with the state-of-the-art streaming algorithms (ThinkDAcc, MASCOT and WRS), BSR-TC always maintains a bounded sampling-ratio to discover more triangles along with the continuous growth of graph streams, and so as to obtains more accurate results and efficient memory usage. Therefore, BSR-TC has adaptive characteristics for triangle counting over evolving graph streams.

A. Experimental Setup

We perform experiments on a server with Intel Xeon Gold 6148 processors and 64-bit Red Hat Linux OS. Each experi-
TABLE II: Summary of the real-world graph streams used in our experiments.

<table>
<thead>
<tr>
<th>Name</th>
<th># Nodes</th>
<th># Edges</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dblp</td>
<td>317,080</td>
<td>1,049,866</td>
<td>Collaboration network</td>
</tr>
<tr>
<td>Skitter</td>
<td>1,696,415</td>
<td>11,995,298</td>
<td>Internet topology graph</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>3,997,962</td>
<td>34,681,189</td>
<td>Friendship network</td>
</tr>
<tr>
<td>Orkut</td>
<td>3,072,441</td>
<td>117,185,083</td>
<td>Online social network</td>
</tr>
</tbody>
</table>

Fig. 4: Scalability. BSR-TC is always more accurate and stable than ThinkDAcc and WRS. Here, We calculate the global error rate every 500,000 edges for evolving graph streams, based on the Orkut dataset.

Fig. 5: Accuracy. More triangles are discovered by BSR-TC, when \( \eta < R \).

### B. Evaluation Metrics

We use the following metrics to evaluate the accuracy of global and local triangle counting, respectively.

- **Global Error.** Let \( \hat{x} \) be the ground truth of the global triangles, and \( x \) be the estimated value of \( \hat{x} \). Considering \( \hat{x} \) may be equal to 0, we add 1 to both \( \hat{x} \) and \( x \). Let \( c \) denote the number of runs for each experiment. Then, the global error is

  \[
  \frac{1}{c} \sum_{i=1}^{c} \left| \frac{\hat{x} - x}{\hat{x} + 1} \right|
  \]

- **Local Error.** Let \( \hat{x}_u \) be the ground truth of the local triangles for each node \( u \in \mathcal{V} \), and \( x_u \) be the estimated value of \( \hat{x}_u \). Considering \( \hat{x}_u \) may equal 0, we add 1 to both \( \hat{x}_u \) and \( x_u \). Then, the local error is

  \[
  \frac{1}{c} \sum_{i=1}^{c} \left\{ \frac{1}{|\mathcal{V}|} \sum_{u \in \mathcal{V}} \left| \frac{\hat{x}_u - x_u}{\hat{x}_u + 1} \right| \right\}
  \]

### C. Performance

Since BSR-TC is a first exploration of triangle counting by adaptively resizing memory budget over evolving graph streams, there are no existing streaming algorithms for similar comparisons. For the sake of illustration, we use ThinkDAcc, MASCOT and WRS as the baselines, which are state-of-the-art streaming algorithms for triangle counting. Here, we define the ratio of initial memory budget against whole graph streams as \( \eta \), and set the threshold \( R \) to 20%.

- **Scalability (maintain accuracy over evolving streams).** In real application scenarios, it is challenging to set appropriate parameters, such as memory budget, for triangle counting, because the scale of graph streams is always increasing as time flies. As shown in Figure 4, when initial memory budget is \( 10^4 \) edges, the global errors of both ThinkDAcc and WRS fluctuate between -0.15 and 0.2 as graph streams evolves, while BSR-TC is almost always equal to zero. This is because the memory budget of ThinkDAcc and WRS is fixed and is difficult to set appropriately without any knowledge about the scale of graph streams. Our proposed BSR-TC can keep stable and high accuracy by the adaptively resampling method over evolving graph streams.

- **Accuracy (regardless of small) \( \eta \).** Figure 5 shows BSR-TC discovers more triangles than ThinkDAcc and MASCOT by adaptively resampling method, when \( \eta \) is less than \( R \). By sampling the first edges in memory budget with probability 1, ThinkDAcc always discovers more triangles than MASCOT. Figure 6 depicts that BSR-TC achieves accuracy of at least 99.8% for global triangles and 60.0% for local triangles, respectively, when \( \eta \geq 0.002 \% \) and \( R = 20\% \). Under this condition, BSR-TC gains accuracy of 100\% for global triangles than ThinkDAcc. Therefore, our proposed BSR-TC can maintain high accuracy by adaptively incrementing memory budget to maintain a bounded sampling ratio \( R \) over the growth of graph streams, even though the initial \( \eta \) is small.

### VI. Conclusions

We propose a single-pass and bounded-sampling-ratio method, BSR-Sample, by adaptively resizing memory budget under control without requiring any prior knowledge about graph streams. BSR-Sample allocates new memory budget to restart a new round of sampling based on standard reservoir
Fig. 6: Accuracy. BSR-TC is more accurate than state-of-the-art streaming algorithms for global and local triangle counting. Here, the $\chi$-axis denotes $\eta$.

VII. ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China (11904370, 61872335, 61732018, 61672499), the Innovation Project of the State Key Laboratory of Computer Architecture (CARC4509), and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing (2019A07).

REFERENCES

Influence Maximization with Consideration of PageRank Centrality and Propagation Probability

Qi Chen, Rong Yan*

Inner Mongolia Key Laboratory of Mongolian Information Processing Technology
College of Computer Science, Inner Mongolia University
Hohhot, China
Email: 1254560848@qq.com, csyanr@imu.edu.cn

Abstract—Influence Maximization (IM) problem has been attracted considerable interest and attention in last decades. However, the centrality algorithm-based methods were with low time complexity but made the acceptability of diffusion vaguely. The main purpose of our work is to select the influential nodes according to the available budget to maximize the impact coverage. Based on the traditional independent cascade model, this paper mainly solves the IM problem, designs two effective PRTH algorithms based on PageRank and propagation probability threshold, and combines PageRank of PRTH processed graph with degree discount algorithm to get an algorithm named PRDD. Experiments on four datasets show that the two algorithms have better performance than the existing algorithms in the aspect of influence diffusion.

Index Terms—influence maximization, PageRank, degree discount, propagation probability

I. INTRODUCTION

Identifying an initial seed set of users who would eventually affect the maximum number of users is the Influence Maximization (IM) problem. It is of great significance in controlling public opinion, rumors and advertising marketing, and it has become a hot research field due to the increase in the number and scale of interaction between users in social networks. At the same time, it also brings massive data to the network, and the value of network data is also increasing. IM was first used in marketing, which is called viral marketing. The spread pattern of viral marketing [1] is based on mutual trust between two users. The purpose of viral marketing is to influence their friends, promote products or pass information to their friends, and then expend this scheme maximization as much as possible, as well as the coverage of information. One of the most active areas of interest in the last decade has been impact maximization and how content or products are adopted by users, which was first introduced in the work of Kempe et al. [2]. Then, in view of the challenge of this problem, kinds of research work have been carried out. Kempe et al. [2] considered IM problem as a combinatorial problem, and proposed an effective greedy algorithm, which can achieve high influence spread among all existing methods, but losing in time complexity. Great efforts followed the work of Kempe et al. [2] and tried to reduce the running time, such as CELF [3] and CELF++ [4]. However, few greedy algorithm-based methods are feasible for large-scale network, while the heuristic method, such as degree discount, instead can achieve an acceptable impact in running time [5].

At present, the existing centrality algorithm-based methods merely use the node degree as a measurement, but ignore the node PageRank. In this paper, we devise a new top-k selection algorithm based on PageRank centrality and propagation probability threshold (PRTH). We calculate the influence diffusion and marginal gain by successive iterative updating. Furthermore, we propose a new PRDD algorithm by combining PRTH with degree discount algorithm to alleviate the effect of the point aggregation. These above algorithms make it possible to calculate the influence diffusion directly and accurately. Through experimental analysis, our algorithm has achieved acceptable results, compared with the current mainstream algorithm in terms of influence diffusion.

The remainder of the paper is organized as follows. Section 2 reviews the related work. Section 3 explains the design and implementation of our methods in detail. Experimental results on four large datasets are provided in Section 4. Finally in Section 5, we conclude this paper.

II. RELATED WORK

IM problem was firstly proposed by Domingo et al. [1] from the perspective of algorithm. Next, Kempe et al. [2] formulated the issue of IM in the social network, and proposed two widely influence diffusion models: Linear threshold model (LT) and Independent Cascade model (IC). As for a non-deterministic polynomial-time (NP)-hard problem, Kempe et al. [2] proposed a greedy approximation algorithm with \( (1-1/E) \) approximation ratio to address this issue. Experimental results showed that the proposed algorithm was more effective than the traditional random methods. However, greedy algorithms in large-scale social networks were inefficient and inaccurate with poor time complexity and memory consumption. The reason mainly is that the greedy approximation ratio method requires tens of thousands of Monte Carlo simulations. In recent years, much efforts were proposed to effectively address IM problem. The related research includes the greedy-based methods and the heuristic-based methods.

Pursuing low time complexity is the question surround the greedy-based algorithms. Lots of researchers tried to improve the greedy-based methods and provide kinds of effective
versions. Leskovec et al. [3] proposed CELF algorithm to
ehance the greedy algorithm by using sub module char-
acteristics in node selection process. The CELF algorithm
could obviously reduce the evaluation times than the simple
greedy algorithm. Goyal et al. [4] proposed a modified CELF
to algorithm, called CELF++, which can significantly reduce the
amount of computation and obtain the better results. Chen
et al. [5] designed a new scheme to improve the greedy
algorithm, and combined their model with CELF algorithm
to get a faster greedy algorithm. Furthermore, they proposed
the prefix of exclusion maximum impact tree by using local
tree model to approximate influence propagation.

Heuristic-based algorithms try to improve the propaga-
effect, and much heuristic-based literatures are based on
centrality. Chen et al. [6] exploited a degree discount
heuristic algorithm, which nearly matched the performance
of the greedy algorithms for the IC model, and improved
upon the pure degree heuristic in the other cascade models.
Nandi et al. [7] proposed a new method called DegGreedy to
maximize the influence spread based on node neighborhoods,
which could provide higher influence spread and good effi-
ciency in terms of scalability. Deng et al. [8] proposed two
centrality-based edge activation probability algorithms under
the IC model, which named NewDiscount and GreedyCIC,
with considering edge probability. Taheri et al. [9] utilized
HellRank centrality measure to identify the most influential
users based on the Hellinger distance between a node pair in
a bipartite graph. Cui et al. [10] proposed a degree-descending
search strategy to obtain all nodes that have the influence
spread as the degree centrality. Lattanzi et al. [11] designed a
random node centrality algorithm based on the phenomenon
of friendship paradox. Mohammed et al. [12] proposed a new
algorithm called PrKa based on Katz centrality. In their work,
the propagation probability threshold permitted to compute
the influence over all the paths and selected the one that
maximizes the influence. Recently, Ding et al. [13] proposed
a new realistic independent cascade (RIC) model and several
greedy maximization algorithms. Maji et al. [14] modified
the k-shell method and compared several variants. Banerjee
et al. [15] proposed a hop-based heuristic method based on
‘expected earned benefit’.

III. METHODOLOGY

Firstly, we give some denotes for the work. We consider
the social network as a undirected graph \( G=(V,E) \), where \( V \)
denotes a group of users, \( E \) denotes a group of relationships.
We represent the number of users in the network as \( |V| \) and
\( |E| \) as the number of edges.

A. Algorithm (PRTH): Pagerank centrality and propagation
probability threshold algorithm.

In this paper, we assume that the larger the PageRank value
is, the more important the node is, and the information is easy
to spread from important nodes to other nodes. But you can
imagine such a situation, when a node connects many edge
nodes with degree 1, its PageRank value will be very high, but
it is not so important. According to the above assumptions,
we solve the problem by the following methods. We set a
weight for each edge and take the average of ownership as
the threshold. Then the edge whose weight is less than the
threshold is removed. This method can effectively remove the
edge nodes, making the distribution of PageRank points more
concentrated in the central area, also for the propagation path
of nodes wider.

Next we will introduce our PRTH algorithm. Firstly, we
construct a undirected graph. Next, we need to get the PageR-
akn value of each node in the graph. Then, we calculate the
edge weight divided by the value of each user with the sum
of the PageRank of the two users connected using the edge.
Through the above description, we can obtain the node pair
weight using equation (1):

\[
weight(u,v) = \frac{\text{pagerank}(u)}{\text{ pagerank}(v) + \text{ pagerank}(u)}
\]  

(1)

Where \( u \) and \( v \) denote two different nodes, respectively.
Through equation (1), we can clearly conclude that the weight
of \( u \) to \( v \) is different from that of \( v \) to \( u \). So here we choose
the maximum weight of each edge pair as the weight of the
each edge pair.

With these steps we get a weighted graph. After that, we
introduce the threshold of propagation probability, which is
reserved only when the weight of the edge reaches this value.
The probability is simply calculated by the average of the
weight of the edges and the number of edges. Therefore,
we obtain the maximum propagation probability threshold \( th \)
using the equation (2):

\[
\text{th} = \frac{\sum_{e}^{|E|} \text{ weight}_{\text{max}}(e)}{|E|}
\]  

(2)

After removing the unqualified edges, we get a new graph.
Next, we will calculate the PageRank value in the new graph
again and arrange it in descending order afresh. Finally, the
top-k node set before sorting are selected as the final seed
node set. We use Algorithm 1 to accomplish it.

B. Algorithm (PRDD): Combining pagerank with degree dis-
count.

Previously, we have concluded that PRTH algorithm can
make the node distribution closer to the center. Unfortunately,
we found that the node aggregation occurs occasionally in the
experiments, and PRTH algorithm may become unstable as
the number of seed nodes increases. Furthermore, we propose
an algorithm named PRDD to modify the situation. In this
algorithm, we fuse the index of PRTH and the index of degree
discount in the original graph, so as to improve the ability of
preventing aggregation.

In this section, we will introduce the influence maximization
model combining PageRank and degree discount. First of all,
we need to calculate the PRTH and the degree discount of the
graph. Secondly, we normalize the two values of PageRank
and degree discount, and then combine the two indicators
through a linear parameter \( \alpha \), which can be computed by
Algorithm 1 PRTH

Input:
original graph \( G=(V,E) \);
seed size \( k \).

Output:
top-\( k \) node set \( S \).
1: page=pagerank(\( G \));
2: for \( i \) in \( G\).edges() do
3: \( w_1 = \frac{\text{page}(t(i))}{\text{page}(i)} \);
4: \( w_2 = \frac{\text{page}(t(i))/\text{page}(i)}{\text{page}(t(i))/\text{page}(i)} \);
5: \( \text{weight}_{\text{max}} = \max(w_1, w_2) \);
6: end for
7: \( \text{th} = \frac{\sum_{i} \text{weight}_{\text{max}}(e)}{\sum_{i} \text{weight}_{\text{max}}(e)} \);
8: for \( i \) in \( G\).edges() do
9: if \( i < \text{th} \) then
10: remove edge \( i \);
11: end if
12: end for
13: \( S \) is the sorted top-\( k \) nodes;
14: return \( S \);

Algorithm 2 PRDD

Input:
\( G=(V,E) \);
PageRank value \( \text{thp} \) in new graph \( G \);
seed size \( k \);
ratio parameters \( \alpha \) of PRDD.

Output:
seed set \( S \).
1: Initial \( S = \phi \) and \( PD = \phi \);
2: for \( i \) in \( G \) do
3: \( d(i) = \text{degree}(i) \);
4: \( dd(i) = 0 \);
5: \( t(i) = 0 \);
6: if \( i \) in \( \text{thp} \) then
7: \( p(i) = \text{th}(i) \);
8: else
9: \( p(i) = 0 \);
10: end if
11: end for
12: \( \text{max degree} = \max(d) \);
13: \( \text{max pagerank} = \max(p) \);
14: for \( v \) in \( G \) do
15: \( PD(v) = \alpha \times \frac{\text{dd}(v)}{\text{max degree}} + (1 - \alpha) \times \frac{p(v)}{\text{max pagerank}} \);
16: end for
17: for \( i \) in \( G \) do
18: \( u = \arg\max(PD|i| \in V \setminus S) \);
19: \( S = S \cup \{u\} \);
20: for neighbor \( v \) of \( u \) and \( v \in V \setminus S \) do
21: \( t(v) += 1 \);
22: \( dd(v) = d(v) - 2 \times t(v) - 0.1 \times t(v) \times (d(v) - t(v)) \);
23: \( PD(v) = \alpha \times \frac{dd(v)}{\text{max degree}} + (1 - \alpha) \times \frac{p(v)}{\text{max pagerank}} \);
24: end for
25: end for
26: return \( S \);

A. Datasets and Experiment Settings

First, we introduce the datasets used in the experiments. We use a social network dataset Facebook, CA-HepTh, CA-HEP-TH (High Energy Physics - Theory) collaboration network is from the arXiv and covers scientific collaborations between authors papers submitted to High Energy Physics - Theory category.

1) Facebook: This dataset of ‘circles’ (or ‘friends lists’) is from Facebook. The data of Facebook is collected from survey participants.
2) CA-HEP-TH: Arxiv HEP-TH (High Energy Physics - Theory) collaboration network is from the arXiv and covers scientific collaborations between authors papers submitted to High Energy Physics - Theory category.
3) NetHEPT: It is an academic collaboration network from arXiv. In this dataset two of them are connected by an undirected link, if they co-authored at least one paper.

As for both above algorithms, we conduct the experiments under the IC model with probability \( p \) set to 0.1. The seed size \( k \) varies from 10 to 50, and the number of iteration is set to 1000.

B. Baseline

In the experiments, five baseline algorithms are compared. The descriptions are presented as below.
1) **Degree** algorithm: This algorithm selects the degree centrality of the top-$k$ propagator with the highest degree centrality.

2) **Pagerank** [16] algorithm: This algorithm counts the number and quality of links from a node to all other nodes to determine the importance of the node. We also choose the first $k$ values as nodes.

3) **Degree Discount** [6] algorithm: This algorithm selects the seed set according to the degree center degree score of nodes, and discounts the edge combined with the next selected seed from the degree calculation of nodes.

4) **PrKatz** [12] algorithm: This algorithm is relied on the use of a combination of Katz centrality and propagation probability threshold tested over each edge for each user in the network. Its parameters are the same as those in the article [12] = 1, $\alpha = 0.0015$.

5) **RIS** [17] algorithm: This algorithm generates reverse reachable set to find the maximum seed nodes.

**C. Evaluation and Analysis**

First of all, we did an experiment to explore the effect of $a$ on the results. We set the value of $a$ from 0.1 to 0.9, and we did experiments on all data sets. In this paper, we present the experimental results on Facebook and NetPHY datasets. We can see that the best value is between 0.1 and 0.3. Similar conclusions can be obtained on the other two datasets.

Next, we compared the baseline method with four datasets as follows:

Figure 3 shows the effect of all algorithms in the Facebook dataset. Compared with other datasets, this dataset has much more edges.

From Fig. 3, we can see that PRDD algorithm performs better in this dataset, while PageRank algorithm performs better merely at the beginning. Generally speaking, these four algorithms, including PageRank algorithm, PRTH algorithm, PrKatz algorithm and RIS algorithm, have similar effect. There is a gap between the Degree Discount algorithm and the above four algorithms, especially when $k$ is set to 10, the Degree algorithm performs the worst. In Facebook, RIS algorithm performs best, which shows that RIS algorithm is more suitable for the dataset with more edges. From Fig. 3, we can see that our algorithm is not much better than other algorithms for the dataset with large average degree.

Figure 4 shows the effect of several algorithms in CA-HepTh dataset. CA-HepTh has the minimum average degree compared with other three datasets.

From Fig. 4, we can see that the effect of PRTH algorithm and PRDD algorithm is similar, and the result of PRDD algorithm is slightly higher than that of PRTH algorithm. Also,
we found that PrKatz algorithm achieved good results in this dataset, and the Degree Discount algorithm follows. At the same time, PageRank, Degree and RIS perform poorly. From this dataset, we can see that PRTH and PRDD have similar effect on relatively small-scale dataset.

Figure. 5 shows the influence diffusion comparison of our algorithms against with the baseline algorithms on NetHEPT. From Fig. 5, we can see that the PRTH algorithm is slightly stronger than the original PageRank algorithm in the initial stage, and the latter two are almost the same. The reason is that there are many nodes with large pagerank value, which makes the PRTH algorithm not play a very important role. The two algorithms appear in the later period, and the growth slowly down, which is consistent with the conclusion that when the number of nodes becomes more, there will be point aggregation. For PRDD algorithm, we can intuitively see that its growth is relatively stable. In the early stage, the effect of PRDD algorithm is slightly worse than PRTH algorithm and original Pagerank algorithm. However, in the later stage, PRDD algorithm has been realized anti super, so we can see that the integration of degree discount and PRTH accomplish a better effect. At the same time, we can also conclude that the PRDD algorithm get a better improvement compared with the original Degree Discount algorithm. The overall effect of the PrKatz algorithm is similar to that of the Degree Discount algorithm. Moreover, we also find that RIS algorithm does not perform well in NetHEPT.

Figure. 6 shows the influence diffusion comparison of our algorithms against with the baseline algorithms on NetPHY. From Fig. 6, we can find that the performance of the original Pagerank algorithm is poor, and the performance of PRTH algorithm works much better. It can be seen that PRTH algorithm plays a great role in NetPHY. But the disadvantage is that when \( k \) is 30, there is a relatively large fluctuation. We have already analyzed that the main disadvantage of PRTH is that it will be node aggregation. From the above results, we can see that there are more node aggregation at \( k \) is 30. On the contrary, PRDD algorithm is more smoother and more effective than PRTH algorithm. Compared with PRDD, the original Degree Discount algorithm has a big gap. RIS algorithm is better than PrKatz algorithm in NetPHY.

Figure. 7 shows the running time of several algorithms on four datasets. Obviously, we can see that PrKatz algorithm runs the longest in four datasets. The reason is that it takes a long time to calculate the Katz centrality at the beginning. The second is RIS algorithm. RIS algorithm takes a long time because it needs to calculate the path. In all datasets, Degree algorithm gains the shortest time. Because Degree algorithm only needs to calculate the degree of each point and sort. Similarly, Pagerank algorithm only needs to calculate the Pagerank value of each point and sort. However, the calculation time of Pagerank is longer than that of degree, but it is much less than that of Katz centrality. So in all datasets, the time of Pagerank algorithm is the third shortest. As shown in Fig. 7, we can conclude that Degree Discount algorithm takes the second shortest time. Compared with Degree algorithm, Degree Discount algorithm needs much calculation cost. But its time-consuming is still shorter than Pagerank algorithm. We can know that there is a big gap between the calculation of Degree and Pagerank. Our algorithms, PRTH and PRDD, rank fourth and third in time-consuming, respectively. PRDD takes a longer time than PRTH, because PRDD combines PRTH and Degree Discount, so the time of PRDD is longer than the sum of PRTH and Degree Discount. Because PRTH algorithm is based on the calculation of Pagerank value, the time cost of it will be longer than Pagerank algorithm.

V. CONCLUSION

In this paper, we exploit two algorithms PRTH and PRDD to tackle IM problem. Our algorithms still provide acceptable results compared to other known methods, and make it possible to calculate the influence spread directly and accurately. Experimental results show the effectiveness of our model that can better influence diffusion. PRTH performs better in moderately degree datasets when there are fewer seed nodes in the network, while PRDD performs better when there are more seed nodes in the network.

ACKNOWLEDGEMENTS

This research is supported by the National Natural Science Foundation of China (Grant No. 61866029) and the Post-
graduate Innovation and Entrepreneurship Foundation of Inner Mongolia University (Grant No. 11200-121024).

REFERENCES


Grasping or Forgetting? MAKT: A Dynamic Model via Multi-head Self-Attention for Knowledge Tracing

Deming Sheng, Jingling Yuan*, Xin Zhang
School of Computer Science and Technology,
Wuhan University of Technology,
Wuhan 430070, China
Email: shengdeming@whut.edu.cn, yjl@whut.edu.cn, xinz@whut.edu.cn

Abstract—The outbreak of the COVID-19 pandemic arises enormous attention to online education then knowledge tracking is an increasingly crucial task with its vigorous development. However, the surge of student historical interactions and the lack of prior knowledge is engendering a sequence of issues, such as the decrease in prediction accuracy while the increase in training time. Simultaneously, most existing approaches fail to provide in-depth insights into why a student is likely to answer the question incorrectly and what affects the knowledge state of the student. To address those issues, we propose a multi-head self-attention model named MAKT for dynamic knowledge tracing, which makes the prediction results interpretable at the model and instance level. The customized multi-head self-attention layer has high training efficiency owing to its parallelization capability and spends about 6 seconds in each epoch on a single GPU. We further visualize the attention weights of MAKT and student knowledge acquisition tracking, finding that not all historical interactions are equally important but the recent interactions profoundly establish the knowledge state of students. In the end, extensive experiments on three datasets demonstrate the robustness and superiorities of MAKT, improving ACC by 1.14% and AUC by 1.20% on average.

Index Terms—MOOC, Knowledge Tracing, Educational Data Mining, Attention Mechanism, Sequence Modeling

I. INTRODUCTION

Online education systems, such as Massive Online Open Course (MOOC), Intelligent Tutoring System (ITS) and Online Judge (OJ) Systems, have a long history dating back to the 1980s [1], [2] and have still witnessed the proliferation with the computer-aid technology and artificial intelligence in recent years. Specifically, students in these systems can finish a series of appropriate tests individually according to their needs and acquire the necessary knowledge in the process of solving relevant exercises. As shown in Fig.1, the availability of such exercising process offers an opportunity to model student learning in terms of predicting student performance (e.g., forecasting whether or not this student can answer an exercise correctly in the next time) and tracking student knowledge state (e.g., estimating the mastery level of key knowledge components based on historical data.).

Knowledge tracing has undergone many paradigm shifts in the past forty years and many approaches have been developed from both educational psychology and data mining areas, such as sparse factor analysis [3], deep learning [4], topic modeling [5] and matrix factorization [6]. Hidden Markov Model (HMM) was traditionally utilized in Bayesian Knowledge Tracing (BKT) and its variants [7]. More recently, a series of Recurrent Neural Network (RNN) based sequential models have been proposed to capture these long term dependencies between the student historical interactions, such as DKT [4] and DKT+ [8]. Simultaneously, Convolutional Neural Network (CNN) is gradually employed to model individualization in the student learning process [9].

Nonetheless, there are still three main challenges in the knowledge tracking task: (1) Long sequence information modelling and (2) Hidden relationship mining between exercises and (3) Interpretation of the prediction results. Existing approaches have achieved certain results in the first two points, but failed to provide in-depth insights into
why a student is likely to answer the question incorrectly and what affects the knowledge state of the student.

In this paper, we propose a Multi-head Self-Attention model for Knowledge Tracing (MAKT). MAKT can effectively improve the predicted performance and dynamically track the knowledge state. More importantly, MAKT has excellent interpretability and the potential to exploit the implicit relationship between exercises without prior knowledge. In summary, our main contributions in this paper are three folds:

- We customize a multi-head self-attention layer to model individualization, positional encoding rather than the traditional RNN-based model is utilized to capture sequence information.
- We perform extensive experiments on three different datasets and demonstrate that MAKT in addition to showing its robustness and superiorities, supports parallel computing.
- We visualize attention weights and student knowledge acquisition tracking, offer intuitive and in-depth insights on the predicted result at both the model and instance level.

II. RELATED WORK

Cognitive diagnosis refers to predict student performance by discovering student states from the exercising records in educational psychology. The traditional cognitive diagnostic models can be divided into two groups: continuous models and discrete models. Taking item response theory (IRT) as an example of the continuous model, IRT utilizes the logistic regression based on the student ability and the exercise (item) difficulty to assume student performance [10]. Discrete models, such as Deterministic Inputs, Noisy-And gate model (DINA), leverage the student knowledge components proficiency by a binary latent vector with a given Q-matrix to improve prediction results [11].

Knowledge Tracing is an essential task for evaluating the knowledge state of a student based on his past interaction. Bayesian knowledge tracing (BKT), followed by Hidden Markov Model (HMM), models the latent knowledge state as a set of binary variables to trace it. Further extensions incorporate more side information about student’s prior knowledge and exercise difficulty into BKT [7]. More recent approaches leverage factorization methods to model individualization with a latent vector that depicts student’s knowledge state [6].

Another line of research includes methods based on Deep Learning, which has achieved great success. Deep Knowledge Tracing (DKT) [4] employs Long Short Term Memory (LSTM) to model student exercising process while DKT+ [8] exploits a regularization term on the foundation of DKT to further improve the predicted performance. Memory Augmented Recurrent Neural Network (DKVMN) [12] is proposed to bridge the gap between exercises and knowledge concepts for a better performance prediction. CKT [9] utilizes a hierarchical convolutional network to model individualization.

III. THE PROPOSED MODEL

A. Problem Definition

In an online education system, suppose there are $S$ students, $E$ exercises, and $K$ knowledge concepts, where students do these exercises individually at different times. As shown in Fig.1, the knowledge tracing (KT) task can be formalized as follows: given a learning sequence $x_s =\{e_1, k_1, r_1, e_2, k_1, r_2, \ldots, e_T, k_K, r_T\}$ or $x_s =\{e_1, r_1, e_2, r_2, \ldots, e_T, r_T\}$ with $T$ learning interactions of a certain student $s$, we aim to assess the knowledge state of students after each learning interaction. Here $e_t$ represents the exercise being answered at learning interaction $t \in T$ and $r_t \in \{0, 1\}$ indicates whether the exercise $e_t$ has been answered correctly (1 stands for right and 0 else). In short, knowledge tracing aims to estimate the probability $P[r_t = 1|e_1, k_1, e_2, k_1, \ldots, e_{t-1}, k_K, r_{t-1}, e_T]$ or $P[r_t = 1|e_1, r_1, e_2, r_2, \ldots, e_{t-1}, r_{t-1}, e_T]$.

In the following, we will specify the probabilistic modelling and parameter learning of MAKT. For better illustration, the key notations are summarized in Table I.

B. Model of MAKT

The framework we propose approach is showed in Fig.2. The core part of our framework is the multi-head self-attention layer, which utilizes the attention mechanism to better model the learning process of students.

Input Embedding. We firstly transform learning sequence of student into an interaction embedding matrix $E^{S \times 2D}$, where $2D$ is the latent dimension. Following [13], we extend the answer value $r_t$ to a zero vector $e_r = (0, 0, \ldots, 0)$ with the same $D$ dimensions as the

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notations</td>
</tr>
<tr>
<td>b</td>
</tr>
<tr>
<td>i</td>
</tr>
<tr>
<td>$r, \hat{r}$</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>Q</td>
</tr>
<tr>
<td>D</td>
</tr>
<tr>
<td>E</td>
</tr>
<tr>
<td>H</td>
</tr>
<tr>
<td>N</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>T</td>
</tr>
<tr>
<td>W</td>
</tr>
<tr>
<td>X</td>
</tr>
<tr>
<td>Q, K, V</td>
</tr>
</tbody>
</table>
exercise embedding \( e_k \) and integrate them into the input embedding as follows:

\[
x = \begin{cases} 
  [e_r \oplus e_k], & \text{if } r_t = 0 \\
  [e_k \oplus e_r], & \text{if } r_t = 1 
\end{cases}
\]  

(1)

Considering the actual situation of different datasets, the original information is distinct, and we adopt two methods to initialize \( e_k \) in this paper. If one exercise relates to one knowledge concept or more, we can construct a binary knowledge matrix \( Q^{E \times K} \). However, the number of feature categories is massive, resulting in the generated matrix being high-dimensional and sparse. Hence, we employ an embedding layer to reduce dimensionality and reshape it into a D-dimensional space \((Q^{E \times K} \sim E^{E \times D})\). The other method is to randomly initialize \( E \) with an embedding layer, for it will be updated automatically in the later training process.

**Position Embedding.** We do not utilize the recurrent and convolution units but the positional encoding to capture sequence information, for the positional encoding is superior in long-distance feature capture capability and operational efficiency. There exist multiple options for position coding [14] and the widely adopted one can be formulated as follows:

\[
PE(t, 2i) = \sin(t/10000^{i/D}) \\
PE(t, 2i + 1) = \cos(t/10000^{i/D})
\]  

(2)

Where \( t \) is the absolute sequence of each interaction and \( i \) is the \( i \)-th dimension of the input embedding \( x \). The adopted positional encoding sine and cosine functions have periodicity. For a fixed-length deviation \( \Delta \), \( PE_{t+\Delta} \) can be expressed as a linear change of \( PE_t \), which is convenient for the model to learn a relative sequence relationship between interactions.

**Self-Attention.** We employ the scaled dot-product attention mechanism [15] rather than additive attention, for this attention mechanism is more computationally efficient and space-saving. The calculation process of self-attention is as follows:

\[
Q = W_Q(x + e_p), K = W_K(x + e_p), V = W_V(x + e_p) \\
Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V
\]  

(3)

where \( Q, K, V \) represent the query, key and value matrix, \( W \in \mathbb{R}^{D \times d} \) is the corresponding weight matrix and \( d_k = D/H \). The purpose of scaling through \( \sqrt{d_k} \) is to avoid too large dot product because when the dot product is too large, the gradient through softmax will be small. And softmax facilitates the gradient calculation of back propagation, and smooth the result to the [0, 1] interval.

**Multi-head Attention.** In order to better satisfy parallel calculation while learning different aspects of attention in different subspaces, the attention weight of \( H \) (Following [15], we set \( H \) to 8) head will be calculated as follows:

\[
\text{Multi-head}(Q, K, V) = \text{Concat}(head_1, ..., head_H)W^O \\
head_j = \text{Attention}(Q_j, K_j, V_j), 1 \leq j \leq H
\]  

(4)

Where \( W^O \in \mathbb{R}^{HD \times d_k} \) is the corresponding weight matrix.

**Add & Norm.** The Add & Norm layer is composed of Add and Norm. Inspired by ResNet [16], Add is a residual connection, usually used to solve the problem of multi-layer network training, allowing the network to focus only on the current difference. Norm refers to layer normalization [17], usually utilized in the RNN structure. Layer normalization converts the input of each layer of neurons into the same mean and variance, which can speed up the convergence. The calculation formula is as follows:

\[
X = \text{LayerNorm}(X + \text{Multi-head}(X)) \\
X = \text{LayerNorm}(X + \text{FFN}(X))
\]  

(5)

**Feed Forward Network.** The encoder block consists of sequentially aligned \( N \) copies (Following [15], we set \( N \) to 6) of encoder layers. A single encoder layer is a multi-headed self-attention layer followed by a feed forward network (FFN) which is defined by:

\[
\text{FFN}(X) = \text{ReLU}(W_1X + b_1)W_2 + b_2
\]  

(6)

where \( W_1, W_2 \) and \( b_1, b_2 \) are weights and biases, respectively. In the end, a fully connected network with sigmoid activation is leveraged to obtain the final probability \( \hat{r}_t \) of the student:

\[
\hat{r}_t = \text{Sigmoid}(WX + b)
\]  

(7)
C. Objective Function

To learn all parameters in MAKT and the input embedding matrix $\mathbb{R}^{S \times 2D}$ in the training process, the objective function is to minimize the negative log likelihood of the observed sequence of student responses. We employ the cross-entropy loss between the prediction $\hat{r}_t$ and actual label $r_t$ with the Adam optimizer [18]:

$$
\mathcal{L} = - \sum_{t=1}^{T} (r_t \log(\hat{r}_t) + (1 - r_t) \log(1 - \hat{r}_t)) \quad (8)
$$

IV. EXPERIMENTS

A. Data Insights

We adopt two real-world public datasets and one synthetic dataset to show the effectiveness of MAKT. Table II shows the statistics of all datasets.

<table>
<thead>
<tr>
<th>DataSet</th>
<th>Students</th>
<th>Concepts</th>
<th>Records</th>
<th>Avg.length</th>
<th>Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSIST2009</td>
<td>4,151</td>
<td>110</td>
<td>325,637</td>
<td>78</td>
<td>0.71</td>
</tr>
<tr>
<td>STATIC2011</td>
<td>333</td>
<td>1,223</td>
<td>189,297</td>
<td>568</td>
<td>0.46</td>
</tr>
<tr>
<td>Synthetic-5</td>
<td>4,000</td>
<td>50</td>
<td>200,000</td>
<td>50</td>
<td>1.00</td>
</tr>
</tbody>
</table>

B. Comparison methods

To illustrate the effectiveness of MAKT, we compare our model with many other models as follows:

- **DKT** [4]: DKT is a deep learning method that utilizes a simple recurrent neural network (RNN or LSTM) to model the exercising process for prediction. We select an LSTM architecture and consider each unique exercise id as a concept associated with the exercise.

- **DKT+** [8]: DKT+ leverages a regularization term based DKT to enhance the consistency in prediction, which effectively alleviates the two problems in DKT. One is that DKT fails to reconstruct the observed input. The other is the predicted performance for knowledge components across time-steps is not consistent.

- **DKVMN** [12]: DKVMN is a Memory Augmented Recurrent Neural Network where in the relation between different knowledge components are assumed by a key matrix and the student proficiency of each knowledge component by a value matrix.

- **CKT** [9]: CKT is a Convolutional Knowledge Tracing method to model individualization. CKT measures the prior knowledge from the historical learning interactions and utilizes a hierarchical convolutional layer to extract individualized learning rates based on continuous learning interactions of students.

C. Evaluation Metrics

For providing robust evaluation results, the performance was evaluated in terms of Accuracy (ACC) and Area Under Curve (AUC), which widely adopted in the binary classification task. Generally, a larger ACC and AUC value demonstrate better performance.

D. Experimental Results

**Student Performance Prediction:** The performance comparison results on three datasets are shown in Table III. We use **bold** to mark the best performance and underline to indicate the best performance other than MAKT. We can observe that MAKT consistently outperform other baseline models on all datasets, which demonstrates the robustness and superiorities of MAKT. Additionally, MAKT gains higher promotions on dataset ASSIST2009 and STATIC2011 with the longer learning sequence length, which indicates that MAKT can capture the core interactions without falling into certain local irrelevant interactions. For the Synthetic-5 dataset, we suspect that a possible reason for the low improvement is that since the number of knowledge concepts in Synthetic-5 is fairly small (five virtual concepts), this hidden relationship between exercises is not distinguishable and MAKT only leverages the sequence relationship modelled by its self-attention mechanism.

**Visualization of Attention Weights:** Benefiting from the attention mechanism, MAKT can offer an intuitive
Table III

Results obtained with different models using there datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>ACC</th>
<th>AUC</th>
<th>%Improv.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DKT</td>
<td>DKT+</td>
<td>DKVMN</td>
</tr>
<tr>
<td></td>
<td>0.7721</td>
<td>0.7734</td>
<td>0.7632</td>
</tr>
<tr>
<td>ASSIST2009</td>
<td>0.8127</td>
<td>0.8129</td>
<td>0.8113</td>
</tr>
<tr>
<td>STATICS2011</td>
<td>0.7511</td>
<td>0.7523</td>
<td>0.7525</td>
</tr>
<tr>
<td>Synthetic-5</td>
<td>0.7820</td>
<td>0.7909</td>
<td>0.7820</td>
</tr>
</tbody>
</table>

Fig. 3. Visualization of attention weights on different datasets.

(a) ASSIST2009  (b) STATICS2011  (c) Synthetic-5

Fig. 4. An example of individualized knowledge tracing result of student.

and in-depth insight on the prediction result with the attention weights visualization, which makes the learning process interpretable at the model level. Fig.3 shows the heatmap of the attention weight matrix on three datasets, each small block depicts the average attention weights of different interaction. An interesting observation is that not all historical interactions are extremely important and the higher weight parameters blocks are concentrated towards the diagonal of the matrix, which can be explained by the forget behaviour rule of the student learning process, that is, the recent interactions profoundly establish the knowledge state of students. Simultaneously, a considerable number of blocks with high attention parameter weights are still scattered in the matrix. Combining with three different datasets, we further find that these interactions share the same knowledge concept with the final interactions, which can be identified by MAKT. A more inspiring conclusion is that the attention mechanism can dig out the hidden relationship between a series of exercises through their attention weights, which benefits the construction of Knowledge Graph in the real world.

Visualization of Knowledge Acquisition Tracking:
To make an intuitive and in-depth insight at the instance level, we visualize the predicted mastery levels (i.e., calculated by Eq.(7)) of an exemplified student with the attached knowledge concepts at each interaction during the exercising process. For better visualization, we filter the six most frequent knowledge concepts rather than distinguishing each specific exercise. As shown in Fig.4, we can notice that the current knowledge state is related to both the original knowledge state and the recent interactions. MAKT can dynamically obtain the knowledge state of the student based on his historical data, which is considered meaningful for further online education auxiliary applications in the real world.

Training efficiency:
Comparing the other baseline methods, the computational efficiency of MAKT is extremely competitive under the same condition. As shown in the Table IV, MAKT only spends about 6 seconds in each epoch on a single GPU which is 11.7 less than the time taken by DKT+, 7 times less than the time

403
taken by DKT, 4.8 times less than the time taken by DKVMN and 2.3 times less than the time taken by CKT. Similarly, MAKT outperforms these models on a single CPU because of its parallelization capability.

V. CONCLUSION

In this paper, we propose a multi-head self-attention based model named MAKT for dynamic knowledge tracing. Specifically, MAKT leverages the historical learning interactions to effectively predict student performance on future exercises and dynamically track the student knowledge state. Simultaneously, MAKT has excellent interpretability and high training efficiency owing to the multi-head self-attention layer, which can offer insights from different levels and support parallel computing. In the end, extensive experimental results demonstrate that MAKT outperforms other baseline models in both ACC and AUC metrics on three different datasets, which indicates the robustness and superiorities of MAKT.

REFERENCES


Deep Similarity Preserving and Attention-based Hashing for Cross-Modal Retrieval

1st Shubai Chen
College of Computer and Information Science
Southwest University
Chongqing, China
chansuba@email.swu.edu.cn

2nd Song Wu*
College of Computer and Information Science
Southwest University
Chongqing, China
songwuswu@swu.edu.cn

3rd Yu Chen
College of Engineering and Technology
Southwest University
Chongqing, China
cy1034429543@email.swu.edu.cn

4th Yuan Yuan
College of Computer and Information Science
Southwest University
Chongqing, China
yy199801@email.swu.edu.cn

Abstract—With the fast progress of deep neural networks and the quick search efficiency of hashing, deep cross-modal hashing (CMH) methods have attracted more and more attention. Generally speaking, the existing CMH methods simultaneously learn hash functions and hash codes in an end-to-end architecture. However, they primarily focus on the hash codes generation stage neglected the losing of rich semantic information in the hash representations learning stage. Besides, the single-label supervision information is leveraged, while most instances are labeled by multiple categories. Thus, we propose a novel Deep Semantic Preserving and Attention-based Hashing (DSPAH) for cross-modal retrieval. In the DSPAH, we first use a cross-level attention block to emphasize significant parts of hash representations and oversee unnecessary ones. Moreover, a Fine-Grained Similarity Criterion (FGSC) is proposed to explore the multiple semantic of image or text instances, helping to learn robust and optimal hash codes. Extensive experiment results on two large-scale public datasets have shown the competition of our proposed DSPAH.

Index Terms—Deep cross-modal hashing, Fine-grained similarity criterion, Cross-level attention

I. INTRODUCTION

Due to the rapid development of search engines and social networks, exponential growth can be seen in multimedia data such as images, text, audio, and video. Thus how to efficiently and effectively retrieve information across these modalities has become a hot spot called multi-modal retrieval. To be specific, one may want to obtain all semantically related instances from the datasets given a text description. However, due to the discrepancies in distribution and inconsistent representations among different modalities, this has raised a significant challenge to unify the gap effectively and efficiently.

Especially, cross-modal retrieval is the most pervasive method of multi-modal retrieval, which aims to map original data (images or text) into similarity preserving embedding in a common latent space [1]. In this way, instances that share similar semantic information may have shorter distances, dissimilar otherwise. The cross-modal retrieval methods can be grossly split into two classes. Traditionally, real-value latent representations is adopted such as [2]–[5]. However, the real-value methods may cause high computational costs and heavy storage burdens. Thus, another popular method called cross-modal hashing (CMH) is proposed to save storage and accelerate the retrieval speed, which leverages Manifold Learning to generate compact hash codes from original high-dimension data.

As the Superior performance of deep learning, Deep Neural Networks (DNN) has shown robust capability in various applications such as [6]–[10]. To take advantages of DNN, many cross-modal hashing methods are proposed including deep cross-modal hashing (DCMH) [11], self-supervised adversarial hashing (SSAH) [12], self-constraint and attention-based hashing network (SCAHN) [13], triplet-based deep hashing (TDH) [14] and multi-label semantics preserving hashing (MLSPH) [15]. However, there are still some issues that need to be solved in the deep CMH community. Firstly, the existing deep CMH methods use a ‘hard’ metric policy to measure the similarity between instances, judged by if two instances share at least one label. However, the simple approximation has neglected the fact that most instances in large-scale cross-modal datasets have multiple labels. Secondly, the hash representations generation and hash codes projection is the equally important part of cross-modal hashing methods. Furthermore, most of the existing deep CMH methods concentrate more on the hash codes generation stage. However, hash representations with less semantic information and spatial relevance may fail to generate optimal hash codes.

A superior Deep Similarity Preserving and Attention-based Hashing (DSPAH) is proposed to solve these problems mentioned above. The framework of DSPAH is illustrated in Fig. 1.

DOI reference number: 10.18293/SEKE2021-050
which corporately learns hash representations and binary codes in an end-to-end architecture. The DSPAH consists of two main components in the hash representations generation stage. CNN model is leveraged to learn rich semantic information from image-modality and text-modality. Moreover, the CNN model is followed by a cross-level attention level where multi-level hash representations are concatenated together as the input. Thus the context relationship and informative information can be obtained by the final hash representations. Moreover, to take advantage of multi-label information, a novel Fine-Grained Similarity Criterion (FGSC) is proposed to build a similarity matrix, which can better explore the semantic relationship among multiple labels.

The core contributions of DSPAH are listed as follows:

- Firstly, a cross-level attention block is proposed to expand intensive semantic information. In this module, hash representations generated from multi-level are concatenated based on the CBAM attention mechanism and further integrated by the adaptive attention matrix, exploring the context correlation and global dependence from both channel and spatial view.
- Secondly, a multi-label preserving calculate criterion called FGSC is proposed to effectively obtain the multi-label information constraint, further generating robust hash codes.
- Finally, the DSPAH is applied on two large-scale cross-modal datasets, and the experimental results illustrate the superiority of our proposed DSPAH compared with other state-of-the-art methods.

The rest of this paper is organized as follows. The detailed description of DSPAH for cross-modal retrieval is presented in section 2. The experimental results and evaluations are illustrated in section 3. Finally, we conclude this paper in section 4.

II. PROPOSED METHOD

A. Problem Definition

We use $G^T$ denotes the transpose of $G$ and $\| \cdot \|_F$ denotes the Frobenius norm. The $\text{sign}(\cdot)$ is an element-wise sign function defined as follows:

$$\text{sign}(x) = \begin{cases} 
1 & x \geq 0 \\
-1 & x < 0
\end{cases} \quad (1)$$

The proposed DSPAH can be expanded to all kinds of modality (e.g. image, text, audio and video) and we mainly concentrate on image-modality and text-modality in this paper. Thus we use $o_i = \{v_i, t_i, l_i\}$ to denote the $i$th training instance, $v_i \in R^{d_v}$, $t_i \in R^{d_t}$ and $l_i \in R^{d_l}$ are image, text and label feature vector with dimension $d_v$, $d_t$ and $d_l$. Moreover, the fine-grained similarity matrix is defined as $S = \{S^{vv}, S^{tv}, S^{tt}\}$, where $S^{vv} = \{S_{ij}^{vv} | i, j = 1, 2, \ldots, N\} \in R^{N \times N}$ and $S^{tt} = \{S_{ij}^{tt} | i, j = 1, 2, \ldots, N\} \in R^{N \times N}$ denotes the intra-modality similarity matrix of image and text, $S^{vt} = \{S_{ij}^{vt} | i, j = 1, 2, \ldots, N\} \in R^{N \times N}$ denotes the inter-modality similarity matrix between image and text.

The most important task of our proposed DSPAH is learning two discriminative hash functions $h^{(v)}(v)$ and $h^{(t)}(t)$ for image-modality and text-modality using the training-set $O$ and similarity matrix $S$. In the hash representations generation stage, hash representations learned from image-modality and text-modality are represented by $F = \{f_{vi} | i = 1, 2, \ldots, N\} \in R^{N \times c}$ and $G = \{g_{ti} | i = 1, 2, \ldots, N\} \in R^{N \times c}$. In hash codes projection stage, $B = \{B_i | i = 1, 2, \ldots, N\} \in R^{N \times c}$ denotes the final hash codes from $F$ and $G$ by simply using a sign function $B = \text{sign}(F + G)$.

B. Network Architecture of DSPAH

The overview architecture of DSPAH is illustrated in Fig. 1, which consists of the multi-level hash representations generation and attention-based interaction module.

Speaking of multiple-level hash representations generation, both the image-network and text-network use Resnet as the bone network because of its remarkable performance on computer vision applications. Especially, the original text data is represented as Bag-of-Words (BoW) vectors and fused into multi-scale BoW representations. To be specific, a multi-scale pooling policy is conducted on the BoW vectors to explore global features, and these vectors are resized into the same length. Furthermore, to facilitate the Resnet [16], these vectors are stacked together to make up a matrix. Therefore, the rich semantics context in text-modality is further explored. For both image-modality and text-modality, we propose cross-level attention to capture the context relationship and global dependence. To be specific, the hash representations from intermediate layers are generated by global average pooling (GAP) and convolution layer with a kernel size of $1 \times 1$. The novel CBAM [17] is leveraged to capture the context relationship and global dependency in intermediate layers. Finally, all of these hash representations are weighted together as the final hash representations by multiplying the adaptive attention matrix. Therefore, the final hash representations can fully obtain the semantic information.

C. Hash Function Learning

In large-scale cross-modal datasets, multi-labels for a single instance(e.g., image and text) are pretty common. However, most previous cross-modal retrieval methods measure the similarity by only one shared label, neglecting the fine-grained similarity among instances. Thus, we propose a new similarity measurement policy called Fine-Grained Similarity Criterion (FGSC) to explore the semantic relationship among instances better. The FGSC of inter-modality can be defined as follows:

$$S_{ij}^{vt} = \frac{l_i^v \cap l_j^t}{\sqrt{l_i^v \times l_j^t}} \quad (2)$$

where $l_i^v$ denotes the label vector of $i$th image instance and $l_j^t$ denotes the label vector of $j$th text instance. $l_i^v \cap l_j^t$ denotes the number of shared labels of vectors $i$th and $j$th.
is the geometric mean of these two label vectors. Similarly, the FGSCs of intra-modality instances are defined as follows:

\[
S^w_{ij} = \frac{t_i^w \cap t_j^w}{\sqrt{t_i^w \times t_j^w}} \quad (3)
\]

\[
S^t_{ij} = \frac{v_i^t \cap v_j^t}{\sqrt{v_i^t \times v_j^t}} \quad (4)
\]

where \(S^w_{ij}\) denotes the similarity across image-modality and \(S^t_{ij}\) denotes the similarity across text-modality. Besides, \(S = \{S^{ut}, S^{vv}, S^{tt}\} \in (0, 1)\). Thus, the hamming-based loss function is no longer suitable for the continuous similarity value. In this paper, the Mean Square Error (MSE) based loss function is adopted to fit the FGSC. Following the common protocol proposed in DCMH, the inner product \(\langle *, *, * \rangle = \langle f, g \rangle\) are leveraged to measure the semantic similarity of hash representations. Therefore, the MSE loss can be defined as follows:

\[
\mathcal{L}_{\text{inter}} = \sum_{i=1, j=1}^{n} \left\| \frac{\langle f_i, g_j \rangle + c}{2} - s_{ij}^{vt} \cdot c \right\|^2 \quad (5)
\]

\[
\mathcal{L}_{\text{intra-image}} = \sum_{i=1, j=1}^{n} \left\| \frac{\langle f_i, f_j \rangle + c}{2} - s_{ij}^{vw} \cdot c \right\|^2 \quad (6)
\]

\[
\mathcal{L}_{\text{intra-text}} = \sum_{i=1, j=1}^{n} \left\| \frac{\langle g_i, g_j \rangle + c}{2} - s_{ij}^{tt} \cdot c \right\|^2 \quad (7)
\]

where \(f_i \) and \(g_j \) are used to denote the hash representations of the \(i\)th image instance and \(j\)th text instance. \(c\) is the length of hash codes. Since the inner product \(\langle *, *, * \rangle \in [-c, c]\), the value range of \(\frac{\langle *, *, * \rangle + c}{2}\) will be the same as \(s_{ij}^{**} \cdot c\).

The purpose of FGSC-based MSE loss is to generate modalspecific and discriminative hash representations \(G\) and \(F\). However, there is a gap between the hash codes and hash representations. Moreover, during the learning procedure of FGSCC-based MSE loss, the similarity between \(B(v) = \text{sign}(F)\) and \(B(t) = \text{sign}(G)\) has been ignored. Since the aim of CMH methods is to learn high-quality hash functions and hash codes, we also need to keep the semantic similarity of \(B(v)\) and \(B(t)\). Another constraint \(B(v) = B(t) = B\) is added to keep the modal invariance. Accordingly, the quantization loss is defined as follows:

\[
\mathcal{L}_q = \frac{1}{c} \left( \|B - F\|_F^2 + \|B - G\|_F^2 \right) \quad (8)
\]

III. Optimization

By assembling the above loss functions, the final overall loss function is given as follows:

\[
\min_{B, \theta_x, \theta_y} \mathcal{L} = \mathcal{L}_{\text{inter}} + \mathcal{L}_{\text{intra-image}} + \mathcal{L}_{\text{intra-text}} + \mathcal{L}_q
\]

(9)

where \(\theta_x, \theta_y\) denote the network parameters of the image-modality and text-modality. An alternating optimization strategy is employed to optimize equation 9. Some parameters will be optimized while others are fixed. The whole optimization algorithm for DSPAH is outlined in Algorithm 1.

IV. Experiment and Discussion

This section evaluates the proposed DSPAH on two large-scale public datasets, MIRFlickr-25K [18], and NUS-WIDE [19] compared with other state-of-the-art methods.

A. Datasets

MIRFlickr-25K [18] is a standard benchmark which contains 25,000 image-text pairs collected from Flickr website.
of different group. Each image is related to several textual descriptions. 20,015 instances of image-text pair with at least one of twenty-four labels are selected, which is similar to DCMH [11]. The text-modality instances are transferred into 1,386-dimensional BoW vectors.

NUS-WIDE [19] The NUS-WIDE includes 268,468 image-text pairs which all belong to 81 categories. A 1,000-dimensional BoW vector is generated for each text-modality instance. In this paper, 190,421 image-text pairs with 21 most common labels have remained, and all instances without supervised information are removed.

We use 10,000 and 10,500 image-text pairs in MIRFLICKR-25K and NUS-WIDE for training. Besides, we stochastically choose 2,000 and 2,100 instances for the query items, and the remained are treated as the retrieval items.

Algorithm 1: Optimization algorithm of DSPAH.

**Input:** Training set \( \{v_i,t_i,l_i\}_{i=1}^{N} \) of intra-modality and inter-modality similarity matrix \( S_{vu}, S_{ut}, S_{vt} \).

**Output:** Optimized parameters \( \theta_x \) and \( \theta_y \) of neural networks and binary codes \( B \).

1. **Initialization:** Initialize the parameters of neural networks, the batch size is set to \( n_v = n_t = 128 \), initialize hash representations of each modality: \( F \) and \( G \), set iteration number \( \text{iter} \) and other hyper-parameters.

2. for \( t=1 \) to \( \text{iter} \) do
   3. Update the parameter \( \theta_x \) of image-network by BP algorithm:
      
      \[
      \frac{\partial L}{\partial f_{ik}} = \sum_{j \in N} (f_j^T f_j + c - 2 \cdot s_{ij}^{vu} \cdot c) \cdot f_{jk} + \sum_{j \in N} (f_j^T g_j + c - 2 \cdot s_{ij}^{vt} \cdot c) \cdot f_{jk} + \frac{2}{c} (F-B)
      \]

   4. Update the parameter \( \theta_y \) of text-network by BP algorithm:
      
      \[
      \frac{\partial L}{\partial g_{ik}} = \sum_{j \in N} (g_j^T g_j + c - 2 \cdot s_{ij}^{ut} \cdot c) \cdot g_{jk} + \sum_{j \in N} (f_i^T g_j + c - 2 \cdot s_{ij}^{vt} \cdot c) \cdot g_{jk} + \frac{2}{c} (G-B)
      \]

5. Update binary codes \( B \)

   \[
   B = \text{sign}(\beta(F+G))
   \]

 Until a fixed number of iterations or convergence;

B. Implementation Details

The DSPAH is conducted on a server with two Nvidia Xp GPU, and the code is written by Pytorch [20] framework. The Resnet-34 with four blocks is utilized to learn rich hash representations. For the image network, the parameters are initialized by the pre-trained model on ImageNet [21]. In terms of the text network, the Normal distribution with \( N(\mu, \sigma^2) \) with \( \mu = 0 \) and \( \sigma = 0.1 \) is leveraged to initialize the parameters. Moreover, pooling sizes of 1, 5, 10, 15, 30 and 50 of BoW vectors are implemented to construct the multi-scale text matrix. We use the SGD as the optimization, and the learning rate is set from \( 10^{-1.5} \) to \( 10^{-6.5} \) on 300 epochs with a mini-batch size of 128.

C. Evaluation and Baselines

To compare the DSPAH with other state-of-the-art methods, we adopt the Mean Average Precision (MAP) and PR Curves to measure the hamming ranking and hash lookup. The details of MAP is defined as follows. Given a query instance \( q \), the Average Precision (AP) is defined as:

\[
AP(q) = \frac{1}{n_q} \sum_{i=1}^{n_{\text{retrieval}}} p_{qi} I(i)
\]

where \( n_q \) is the number of semantic similar instances of query instance \( q \) in database, \( n_{\text{retrieval}} \) is the number of total instances in database. \( p_{qi} \) indicates the probability of instances of top \( i \) instances in retrieval set being similar to the query \( q \). \( I(i) \) is an indicator function, where \( I(i) = 0 \) denotes the \( i \)th instance is dissimilar to the query \( q \), \( I(i) = 1 \) otherwise. For the \( n_{\text{query}} \) instances, the Mean Average Precision (MAP) is defined as follows:

\[
MAP = \frac{1}{n_{\text{query}}} \sum_{j=1}^{n_{\text{query}}} AP(q_j)
\]

Several baseline methods are compared with DSPAH including CMSSH [22], SCM [23], GSPH [24], DCMH [11], CMHH [25], PRDH [26], CHN [27], SepH [28] and SSAH [12]. The MAP results is illustrated in Table I and the PR Curves is demonstrated in Fig. 2 and Fig. 3. From the results, we can get the following observation.

- The DSPAH significantly outperforms other state-of-the-art methods on 16, 32, 64 bits of hash codes in terms of MAP and PR Curves, which clearly shows its superiority. The advance of DSPAH is partly because the cross-level attention dramatically improves the hash representations of interest to concentrate on the vital part and ignore the unconsidered ones.
- The SSAH and DSPAH surpass other deep architecture-based CMH methods and show competitive results, which indicates the importance of preserving multiple semantic labels. The FGSC we proposed in this paper may have the ability to unify the inter-and intra-modality heterogeneity.
- Deep CMH methods such as DCMH, CMHH, SSAH, CHN, and PRDH distinctly attain better performance than other shadow-based CMH methods, including CMSSH,
<table>
<thead>
<tr>
<th>Method</th>
<th>Image query Text</th>
<th>Text query Image</th>
<th>Image query Text</th>
<th>Text query Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMSSH [22]</td>
<td>0.5600 0.5709 0.5836</td>
<td>0.5726 0.5776 0.5753</td>
<td>0.3092 0.3099 0.3396</td>
<td>0.3167 0.3171 0.3179</td>
</tr>
<tr>
<td>SCM [23]</td>
<td>0.6354 0.5618 0.5634</td>
<td>0.6340 0.6458 0.6541</td>
<td>0.3121 0.3111 0.3121</td>
<td>0.4261 0.4372 0.4478</td>
</tr>
<tr>
<td>SePH [28]</td>
<td>0.6740 0.6813 0.6830</td>
<td>0.7139 0.7258 0.7294</td>
<td>0.4797 0.4859 0.4906</td>
<td>0.6072 0.6280 0.6291</td>
</tr>
<tr>
<td>DCMH [11]</td>
<td>0.7316 0.7343 0.7446</td>
<td>0.7600 0.7737 0.7805</td>
<td>0.5415 0.5597 0.5614</td>
<td>0.6011 0.6014 0.6114</td>
</tr>
<tr>
<td>CHN [27]</td>
<td>0.7504 0.7495 0.7461</td>
<td>0.7776 0.7775 0.7798</td>
<td>0.5754 0.5966 0.6015</td>
<td>0.5816 0.5967 0.5992</td>
</tr>
<tr>
<td>PRDH [26]</td>
<td>0.6952 0.7072 0.7108</td>
<td>0.7626 0.7718 0.7755</td>
<td>0.5919 0.6059 0.6116</td>
<td>0.6155 0.6286 0.6349</td>
</tr>
<tr>
<td>SSAH [12]</td>
<td>0.7745 0.7882 0.7990</td>
<td>0.7860 0.7974 0.7910</td>
<td>0.6163 0.6278 0.6140</td>
<td>0.6204 0.6251 0.6215</td>
</tr>
<tr>
<td>CMHH [25]</td>
<td>0.7334 0.7281 0.7444</td>
<td>0.7320 0.7183 0.7279</td>
<td>0.5530 0.5698 0.5924</td>
<td>0.5739 0.5786 0.5889</td>
</tr>
<tr>
<td>DSPAH</td>
<td>0.7978 0.8097 0.8179</td>
<td>0.7902 0.7946 0.8115</td>
<td>0.6498 0.6787 0.6834</td>
<td>0.6396 0.6529 0.6792</td>
</tr>
</tbody>
</table>

**TABLE I**
Mean Average Precision (MAP) comparison results

**Fig. 2.** Performance on MIRFlickr-25K evaluated by PR Curves

GSPH, SCM, and SePH. This reveals the robust and advanced character of deep neural networks, obtaining richer semantic information than the hand-crafted features. Therefore, better results can be observed.

**V. CONCLUSION**

In this paper, cross-level attention and a Fine-Grained Similarity Criterion (FGSC) are proposed, with the vision of learning context-relevant hash representations and generating optimal hash codes. Besides, the attention mechanism can better enhance the ability to focus on the image’s and text’s ‘right’ area. Evaluations conducted on two datasets demonstrate the significant performance of DSPAH compared with other CMH methods. In the future, we are going to use different metrics to investigate the similarity of embeddings.

**ACKNOWLEDGMENT**

The authors appreciate helpful comments from the reviewers on improving our work. This work was supported by the NSFC(61806168), Fundamental Research Funds for the Central Universities (SWU117059), and Venture & Innovation Support Program for Chongqing Overseas Returnees (CX2018075).

**REFERENCES**

Fig. 3. Performance on NUS-WIDE evaluated by PR Curves


An Efficient ROS Package Searching Approach Powered By Knowledge Graph

Long Chen1,2, Xinjun Mao1,2*, Yinyuan Zhang1,2, Shuo Yang1,2, Shuo Wang1,2
1College of Computer, National University of Defense Technology, Changsha, China
2Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha, China
{chen_long, xjm, yinyuanzhang, yangshuo11, wangshuo15} @nudt.edu.cn

Abstract—Over the past several years, the Robot Operating System (ROS), has grown from a small research project into the most popular framework for robotics development. It offers a core set of software for operating robots and that can be reused on different hardware platforms. With thousands of packages available per stable distribution, encapsulating algorithms, sensor drivers, etc., it is the de facto middleware for robotics. However, finding the proper ROS package is a nontrivial task because ROS packages involve different functions and even with the same function, there are different ROS packages for different tasks. So it is time-consuming for developers to find suitable ROS packages for given task, especially for newcomers. To tackle this challenge, we build a ROS package knowledge graph, ROSKG, including the basic information of ROS packages and ROS package characteristics extracted from text descriptions, to comprehensively and precisely characterize ROS packages. Based on ROSKG, we support ROS packages search with specific task description or attributes as input. A comprehensive evaluation of ROSKG shows the high accuracy of our knowledge construction approach. A user study shows that ROSKG is promising in helping developers find suitable ROS packages for robotics software development tasks.

Index Terms—Knowledge Graph, ROS package searching, NLP

I. INTRODUCTION

Writing software for robots is difficult, particularly as the scale and scope of robotics continue to grow. As a framework for building robotics software, ROS is designed with the promise of making development easier through modular design and code reuse1. It offers an abstraction layer between the hardware and application layers, providing hardware manipulation primitives that hide the heterogeneity of the underlying hardware, as well as helping manage the communication between robots.

ROS also provides a package management system to simplify code reuse, so developers can contribute their own applications back to ROS in the form of packages. It is widely used by robotics developers and contains 6,191 packages across its 13 distributions2. The ROS architecture and package system have led to the success of ROS: ROS is considered as the de facto standard for robot programming [4].

When developers need a new feature, developers use a search engine to look for an existing package that implements the feature. But for beginners, since they do not know the ecosystem well, they may choose the inappropriate ROS package during the reuse process, causing the task to fail [4]. For ROS-based robotics software development, ROS Wiki3 is the most commonly used knowledge search community, especially for newcomers. It provides the most basic keyword search function to meet the most basic requirements for developers to find knowledge to solve tasks. Like many other search engines, what it returns are related web links, not direct answers to the task, so developers need to find relevant software packages from the detail page of these web links. Moreover, it can return a limited number of links (Related ROS packages, Q&A posts, tutorials), and requires the developer to click on the link and read the details page to obtain relevant information. Once the searched keywords are not included in the relevant ROS package, it becomes difficult to retrieve and developers need to constantly adjust according to the existing retrieval information, which is time-consuming. So the current search for ROS package knowledge does not meet the needs of developers well.

In this paper, we focus on extracting the rich semantic expression of ROS package and its related information(e.g., related dependencies and messages, etc.). Based on this, we can more effectively characterize the ROS package to better support the recommendation of the related ROS package. Specifically, we firstly design a web crawler framework to obtain ROS package descriptions and some structured information (e.g., sensor, motor and robot which ROS package belongs to). Then we use natural language processing methods to parse the description text to obtain more fine-grained features. In order to make up for the lack of description information, we extract and analyze features from the ROS package names. Finally, we apply our approach to the Kinetic distribution of the ROS packages, and obtain 25,484 entities and 62,854 relationships. All the above information serves as the foundation for constructing a comprehensive Knowledge Graph of ROS package (ROSKG) to enable efficient ROS

DOI:10.18293/SEKE2021-063
* Corresponding author.
1 http://wiki.ros.org/ROS/Introduction
2 https://index.ros.org/stats/
3 http://wiki.ros.org/
package search. We conduct two experiments to evaluate our approach. A comprehensive evaluation of ROSKG shows the high accuracy of our knowledge construction approach. A user study shows that ROSKG is promising in helping developers find suitable ROS package for robotics software development tasks.

Our main contributions are summarized as follows:

- We introduce the idea of using knowledge graph based on semantics representation of package information for the task of ROS package searching. To the best of our knowledge, this is the first study to build a knowledge graph of ROS package and address the problem of ROS package selection.
- We leverage techniques of relation linking and text processing to convert semi-structured and unstructured ROS-related knowledge into a knowledge graph, and develop a search engine, which uses natural language as query input, to solve ROS package searching problems.
- We evaluate the quality of the key steps for ROS package knowledge graph construction and the usefulness of the knowledge graph on ROS package searching.

The rest of the paper is organized as follows. In the next section, we review some related works. The details of our approach are presented in Section III. We provide the evaluation of our work in Section IV. We discuss the threats of validity in Section V. Finally, we conclude our work in Section VI.

II. RELATED WORK

A. ROS

In recent studies, researchers pay more attention to the ecology of ROS and the dependencies between ROS packages, etc. Pichler et al. studied the interdependencies between ROS packages on GitHub, BitBucket, and the rosdistro, and how quality propagates through the dependency network [10]. In an empirical study consisting of interviews and a survey with ROS developers, Estefo et al. investigated the difficulties that ROS users encounter when reusing ROS packages, main contribution bottlenecks in ROS ecosystem [4]. In a separate prior study, Estefo et al. studied code duplication in ROS packages [3]. Alami et al. conducted a qualitative study to better understand quality assurance practices within the ROS community [1]. Kolak et al. focused on ecosystem structure, collaboration, code reuse, and ecosystem health. They found that the most widely used ROS packages belong to a small cluster of foundational working groups (FWGs) [6].

The above is mainly concerned with the problems of ROS ecosystem and the status quo of ROS packages reuse but does not involve how to better realize ROS packages reuse.

B. Knowledge Graph in Robotics Development

Recently, knowledge graphs as a form of structured knowledge have drawn great research attention from both the academia and the industry [5], but few researchers have studied the knowledge graph about robotics development. Zamanirad et al. designed a bot programming platform that dynamically synthesizes natural language user expressions into API invocations and constructed an API knowledge graph to encode and evolve APIs to help robot understand the natural language spoken by humans [14]. Although their knowledge graph is applied in the field of robot, it is not specific to the robotics software development, what we are more concerned about.

III. METHODOLOGY

We propose a knowledge graph based approach to overcome the barriers mentioned above. Fig. 1 presents the key steps in our approach, which contains two main parts: mining ROS package knowledge graph from official website page information and searching ROS package based on the mined knowledge graph. We first use the popular web crawler tool Scrapy\(^4\) to crawl structured content and text description. Then we extract package-related knowledge from structured content and establish connections, including the relevant content of the software package what it provides (launch file, service, plugin and message, etc), metapackage (A set of ROS packages related to a certain function) and the hardware to which it belongs (sensor, motor or robot, etc). In order to get more characteristics information about ROS packages, we use natural language processing methods to obtain category and functionality from the description text to express ROS packages more abundantly. Since the description information of the ROS package is not very complete, and some even do not have it at all, therefore we extract features from the package name. At last, we build a knowledge graph called ROSKG based on the knowledge above for retrieval.

A. ROS Package Characteristic Extraction

Category and functionality: In order to dig out more information about the ROS package, we parse the package description sentence to extract the category and functionality of the package. We first use Stanford CoreNLP tool [9] to obtain the Part-of-Speech(POS) tags of the description sentence. Tokenization is used to break the text into words, phrases, or symbols. POS will represent the category of words which has similar grammatical properties. Then we

\(^4\)https://scrapy.org/
use rule-based chunking technique [15] to chunk category and functionality of the package. In our system, the category and functionality are identified by the regular expressions as shown in Table I. The terms in the table have the same meaning as [15]. Specifically, we extract verb (or verb phrase) followed by noun (or noun phrase) as functionality and noun (noun phrase) as category. We stipulate that noun and noun phrase in functionality do not belong to category. For example, The description of ROS package “grid_map_ros”, “ROS interface for the grid map library to manage two-dimensional grid maps.”, we chunk category “ROS interface” and “the grid map library”; functionality “manage two-dimensional grid maps” as shown in Fig. 2.

**TABLE I: Regular Expression of Different Chunks**

<table>
<thead>
<tr>
<th>Name</th>
<th>Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>(MD)<em>(VB.</em>)+(CD)<em>(DT)</em>+(CD)<em>+(JJ)</em>+(CD)*</td>
</tr>
<tr>
<td>Category</td>
<td>(CD)<em>(DT)</em>+(CD)<em>+(JJ)</em>+(CD)<em>(VBD)</em>(VBG)+(NN.<em>)</em>+(POS)<em>(CD)</em>(VBD)<em>(VBG)</em>(NN.<em>)</em>+(POS)<em>(CD)</em>(NN.<em>)</em>+</td>
</tr>
</tbody>
</table>

**Feature:** Not all ROS packages have description information, which are even incomplete or missing. It is not enough to rely on description information to characterize the ROS package. We notice that the words that make up a package name can well reflect its characteristics. For example, `turtlebot3` in the package `turtlebot3_navigation` reflects this package belonging to the robot `turtlebot3`, and `navigation` reflects that it is a package of the navigation type.

After the above analysis, we extract the last word of the package name of all ROS packages and analyze. However, there are many words that are just different in expression, but the actual meaning is the same in the robot software development process. So we classify words with the same meaning into one category and use one of the words to represent them. For example, “msg” is the abbreviation of “message”, “message” is the singular form of “messages”, so we use “message” to represent them. After the above processing, we select words that appear more than 5 times and analyze their actual meaning. Then we give their definitions. As shown in Table II, we show the most frequent words (top 10) and their definitions.

**B. Linking Entity to ROS Package**

Since the category and functionality of ROS package is originally extracted from the description of ROS package, the connection is naturally established. In this phase, we mainly introduce how to link the extracted entities to ROS package, including package feature introduced in Section III-A and hardware entities.

**Hardware linking:** Through the analysis of the package name above, we find that in general, the first word of the package name is likely to represent hardware information. So we construct an entity dictionary, which contains all hardware words and we match hardware entities crawled from ROS Wiki with the dictionary to establish the linking. In order to more accurately match and establish relationship with ROS package, we have also sort out synonyms for hardware entities. For example, “Velodyne HDL-64E 3D LIDAR” is often referred to simply as “Velodyne”. Furthermore, we take the hardware dictionary to match the description to establish potential relationship.

**Feature linking:** According to the source of the feature, it is mainly reflected in the package name. But we find that some packages contain more than one feature, such as “ainstein_radar_gazebo_plugins”, which includes gazebo and plugin feature. So we split the package name according to the underscore and use the constructed feature dictionary to match to establish a relationship with ROS package. Hardware entities are generally more domain-oriented vocabularies, and feature entities are some commonly used vocabularies with a higher frequency, so hardware can establish potential relationships through description information. If the feature entities do the same, it may establish many relationships that don’t
IV. EVALUATION

We conduct evaluations to explore the following research questions:

- **RQ1**: How is the intrinsic quality of the knowledge captured in the constructed ROSKG?
- **RQ2**: How does ROSKG perform in ROS package searching task compared with ROS Wiki engine?

A. RQ1: Quality of The Constructed ROSKG

The knowledge is extracted from two main sources: structured document content and textual description. Since knowledge extracted from structured information is intrinsically accurate, we mainly evaluate the accuracy of the knowledge extraction from text (i.e., functionality extraction, category extraction, feature linking).

1) Protocol: Similar to previous studies [8] [11] [13], we adopt a sampling method [12] to ensure that the ratio observed in the sample is within 5 confidence interval, and is extended to the population at 95% confidence level. We randomly select 349 functionalities, 369 categories, and 342 feature links to conduct the experiment.

Two developers (who are not involved in this study and familiar with ROS) independently perform the examination. All decisions are binary (the accuracy rates are Acc1 and Acc2 respectively). For the data instances that the two annotators disagree, they have to discuss and come to a consensus. We compute the final accuracy after resolving the disagreements (AccF) and compute Cohen’s Kappa agreement (Kap.) [7] to evaluate the inter-rater agreement. Based on the consensus annotations, we evaluate the quality of the created knowledge about ROS package.

2) Results and Analysis: The results are shown in Table III. We can see the agreement rate are all above 0.78, indicating substantial or almost perfect agreement. The accuracy is generally high (above 0.91) except for the category extracted from description sentences (above 0.73).

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Acc1</th>
<th>Acc2</th>
<th>AccF</th>
<th>Kap.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>92.3%</td>
<td>91.7%</td>
<td>92.0%</td>
<td>0.78</td>
</tr>
<tr>
<td>Category</td>
<td>77.0%</td>
<td>73.4%</td>
<td>75.3%</td>
<td>0.80</td>
</tr>
<tr>
<td>Feature Linking</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Typical problems of ROS package characteristics extraction include: 1) POS tagging or dependency parsing error, e.g., “This package” from sentence “This package contains numerous examples of how to use SMACH” is tagged as a noun; 2) meaningless characteristics, e.g., “is a ROS-Package” from sentence “This is a ROS-Package for libviso2 a library for visual odometry..” is extracted as a functionality of the ROS package “libviso2”; 3) incomplete sentences caused by incorrect HTML parsing or sentence splitting, e.g., “The move_base package provides an implementation of an action (see the”); 4) overly simplistic description sentence, e.g., description sentence just repeats the ROS package name, but is extracted as a category.

---

Fig. 3: Conceptual Schema of ROS Package Knowledge Graph

The other relationship can be naturally established by structured data, such as dependency, metapackage which the package belongs to. These relationships can establish relationships between packages that seem to be unconnected on the surface, for example, the ROS package “turtlebot3_navigation” depends on the other ROS package “amcl”.

C. ROSKG Empowered ROS Package Search

For a specific ROS package searching, the ROSKG can group the ROS package that have the same attributes, e.g., sensor, robot, metapackage, etc, so developers can perform search tasks based on these attributes and view the related content, such as service, message, etc.

For ROS package searching related to specific tasks, given a query phrase or sentence in natural language description, which may not involve entities such as package names, we first use the parsing techniques mentioned in Section III-B to detect functionality and category in the query. We use the start-of-the-art pre-trained word embedding BERT [2] to separately vectorize the components in the query and the components in the description sentence in the knowledge graph. The functionalities and categories extracted from the description sentence have been vectorized in advance. We then compute the phrase similarity by the cosine similarity of phrase embedding. After matching the functionalities and categories compartments of the query sentence and the components in the ROSKG, we rank the ROS package in the ROSKG by the sum of the similarity of the matched counterparts. Finally we give top 10 ROS packages according to the similarity under the premise that above the user-specified threshold (generally 0.8).

D. Proof-of-Concept Implementation

We apply our knowledge graph construction methods described in Section III-B and Section III-C to the Kinetic distribution and use Neo4j\(^3\), a graph database, to construct the knowledge graph for the ROS package. The resulting ROSKG consists of 25,484 entities and 62,854 relationships. Related concepts and their relations can be explained by the conceptual schema shown in Fig. 3.

\(^3\)http://neo4j.com/
The feature linking have 100% accuracy, which is unsurprising because the features are extracted form package name and manually checked and filtered. So the feature linking can maintain a better result.

Our ROS package characteristics and relationship extraction methods for constructing ROS package knowledge graph are basically accurate, which can support practical use.

### B. RQ2: Usefulness Evaluation

We evaluate the usefulness of ROSKG in ROS package searching tasks, that is, choosing the most suitable ROS package for the specific task.

1) **Task:** We extract the main tasks from the book “ROS Robotics Projects” written by Lentin Joseph, which is an introductory book for ROS learners and has a high authority. As shown in Table IV, the book mentions 11 ROS robotics development tasks, each task involves several subtasks, each of which contains 1-2 ROS packages. Finally, we summarize 18 subtasks, involving 21 ROS packages. The participants can formulate any query they wish based on the search task descriptions and the hints from previous search results.

2) **Baseline:** We use ROS Wiki’s search engine as the baseline tool. For ROS-based robotics development, ROS Wiki is the most commonly used knowledge search community, especially for newcomers.

3) **Protocol:** We recruit 6 master students from our school and all of them have almost no ROS-based robotics software development experience. We believe these students are qualified for our study. Furthermore, they also simulate the target audience that our tool aims to assist, i.e., developers who may lack relevant knowledge in finding suitable ROS packages for the specific task. Then we randomly allocate them into two equivalent groups: the control group uses the ROS Wiki’s search engine (P1-P3), while the experimental group uses our ROSKG to complete the tasks (P4-P6).

4) **Results and Analysis:** Table V shows the average task completion time and the number of correct answers by each participant in the two groups. We can see that two groups have the similar answer correctness but the experimental group complete the task faster with narrower standard deviation than the control group. The participant in the experimental group completed the tasks 52.0% faster (55.2 seconds on average) than the control group.

Through interviews with participants, we know that the control group participants often have to scroll the document back and forth in ROS Wiki and compare several documents to pinpoint and cross-validate the function of ROS packages. In contrast, the experimental group participants can view ROS package information in a more structured way, which can help them understand the function of the ROS package faster. We also look into the correct answer rate for each task, for some tasks, both search engines perform well, such as Task#6/11. But neither our search engine nor ROS Wiki performs well on Subtask#5-1. That’s because the information about the ROS

<table>
<thead>
<tr>
<th>Task</th>
<th>Subtask</th>
<th>Related package</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.Use ROS, opencv and dynamixel servos for face detection and tracking</td>
<td>1-1 driver for V4L USB camera</td>
<td>usb_cam</td>
</tr>
<tr>
<td></td>
<td>1-2 servo motor for dynamixel</td>
<td>dynamixel_motor</td>
</tr>
<tr>
<td></td>
<td>1-3 opencv</td>
<td>vision_opencv</td>
</tr>
<tr>
<td>2.Build a chatbot like Siri in ROS</td>
<td>2-1 to translate commands</td>
<td>sound_play</td>
</tr>
<tr>
<td>3.Use ROS to control embedded circuit boards</td>
<td>3-1 ROS for arduino platforms</td>
<td>rosserial_arduino</td>
</tr>
<tr>
<td>4.Operate the robot remotely using gestures</td>
<td>4-1 turtlebot simulation</td>
<td>turtlebot_gazebo</td>
</tr>
<tr>
<td>5.Object detection and recognition</td>
<td>5-1 object detection and recognition</td>
<td>find_object_2d</td>
</tr>
<tr>
<td>6.Use ROS and TensorFlow for deep learning</td>
<td>6-1 convert ROS message to OpenCV image data type</td>
<td>cv_bridge</td>
</tr>
<tr>
<td>7.Run ROS on MATLAB and Android</td>
<td>7-1 Android development package</td>
<td>cv_camera</td>
</tr>
<tr>
<td>8.Building an autonomous mobile robot</td>
<td>8-1 to generate maps</td>
<td>android_core</td>
</tr>
<tr>
<td>9.Use ROS to create self-driving cars</td>
<td>9-1 Velodyne HDL-64E 3D LIDAR</td>
<td>rosjava</td>
</tr>
<tr>
<td>10.Use VR headsets and Leap Motion to remotely control robots</td>
<td>10-1 gesture sensor</td>
<td>map_server</td>
</tr>
<tr>
<td></td>
<td>10-2 visualization tool</td>
<td>velodyne</td>
</tr>
<tr>
<td></td>
<td>11-1 websocket interface</td>
<td>leap_motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rviz</td>
</tr>
<tr>
<td>11.Control the robot through the network</td>
<td>11-2 to set and publish joint state values for a given URDF</td>
<td>rosbmage</td>
</tr>
<tr>
<td></td>
<td>11-3 HTTP Streaming of ROS Image in Multiple Formats</td>
<td>joint_state_publisher</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table V: Accuracy of ROS Package Knowledge</th>
<th>Ave Task Time(seconds)</th>
<th>#Correct Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experimental Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P1</td>
<td>55.4</td>
<td>13</td>
</tr>
<tr>
<td>P2</td>
<td>53.9</td>
<td>14</td>
</tr>
<tr>
<td>P3</td>
<td>56.4</td>
<td>16</td>
</tr>
<tr>
<td>Ave±stddev</td>
<td>55.2±1.0</td>
<td></td>
</tr>
<tr>
<td>Control Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>79.8</td>
<td>12</td>
</tr>
<tr>
<td>P5</td>
<td>83.6</td>
<td>12</td>
</tr>
<tr>
<td>P6</td>
<td>88.2</td>
<td>14</td>
</tr>
<tr>
<td>Ave±stddev</td>
<td>83.9±3.4</td>
<td></td>
</tr>
</tbody>
</table>
package “find_object_2d” cannot be obtained from our query statement.

ROS Wiki can return very relevant online documents for user queries, including tutorials, Q&A post and documents, but not everyone can find the correct ROS package through these resources. For example, the answer to Subtask#8-1 is hidden in a Q&A post, P6 finds it but P4&P5 don’t.

We find that the participants have difficulty in choosing since there are many similar packages related to the task. For example, P5 searches both “rviz” and “octovis” for Subtask#10-2. The relevant information for the two result actually contain “visualization tool”, but “octovis” is a specialized tool for “OctoMap”, which may not meet the current task. P3 makes the similar mistake when carrying out the Subtask#1-2.

One of the biggest problems with ROS Wiki search is that its search method is only keyword matching and only show one page. Once the searched keywords are not included in the relevant ROS package, it becomes difficult to retrieve and time-consuming. For example, participants in control group all return the wrong answer for Subtask#2-1, because the information of the highest ranked ROS package only contains keywords “commands” and “translate” separately, not phrases “translate commands”. But for our search engine, we use the fuzzy query method, even if it is not able to match keywords completely, we can return the relevant software package based on the similarity.

Although by no means conclusive due to the small-scale of our study, our pilot user study demonstrates that our approach significantly decreases the amount of time developers need for ROS package search tasks.

V. Threats to Validity

A threat to internal validity is that some software packages do not have descriptive information, that is to say, no textual information is provided, which will make it impossible to link to the corresponding software package via natural language. Another threat to internal validity is that our database is not complete enough to include all distributions of ROS packages. In the future, we will continue the collection work through automatic methods, which will contribute to further development.

The major threat to external validity is the generalization of our results and usefulness study is small scale. In the future, we will reduce this threat by applying our approach to more open tasks related to robotic software development and release our knowledge graph for public evaluation.

VI. Conclusion

In this paper, we propose an efficient ROS package search approach based on knowledge graph. We leverage advanced NLP techniques for extracting the rich characteristics to better represent ROS packages. Our evaluation confirms the quality of different kinds of knowledge in the knowledge graph, and the usefulness of the generated ROS package search results. In the future, we will refine text processing techniques and design more rules to select meaningful characteristics to improve and extend our approach.

ACKNOWLEDGEMENT

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1004202.

REFERENCES

KatGCN: Knowledge-Aware Attention based Temporal Graph Convolutional Network for Multi-Event Prediction

Xin Song, Haiyang Wang, Kang Zeng, Yujia Liu, and Bin Zhou
College of Computer, National University of Defense Technology
Changsha, China
Email: {songxin, wanghaiyang19, zengkang, liuyujia, binzhou}@nudt.edu.cn

Abstract—Social events are due to gradually changing relations between entities including citizens, organizations, and national governments. Predicting multiple co-occurring events of different types in the future can help analysts understand social dynamics better and make quick and accurate decisions in advance. However, due to the overlook of the knowledge (e.g., event actors and different relations between them), existing methods are insufficient to model the structural and temporal dependence of events with different types simultaneously to better realize the prediction of future multiple co-occurring events. In the paper, we propose a novel Knowledge-aware attention based temporal Graph Convolutional Network (KatGCN) for predicting multiple co-occurring events of different types. We model social events as temporal event graph and extract static features (e.g., event back- ground, topic keywords) from event content to enhance semantic of event graph. We design knowledge-aware attention based graph aggregation method to capture the structure dependence of co-occurring events with different types. We apply temporal encoding to capture the temporal dependence between temporally adjacent events. Empirical results on five-country datasets show that KatGCN outperforms state-of-the-art methods. Further studies verify the effectiveness and interpretability of our model.

Index Terms—multi-event prediction, knowledge-aware attention, temporal Graph Convolutional Network

I. INTRODUCTION

Social events such as protests, cooperation, and fights occur frequently and have a significant impact on society. It is highly desirable to predict multiple co-occurring events of different types, aka, multi-event, in advance to reduce the potential social upheaval and damage caused. Prior work [1] [2] mainly focused on predicting the scale of events or whether a given-type event will occur in the future. They have achieved good performance in the prediction of given-type events. However, as for multi-event prediction, they ignore the potential dependence of multiple co-occurring events given that different models are trained for different types of events.

Currently, social events are often extracted from news articles and structured as temporal knowledge graph with additional textual features [3], also called temporal event graph. As shown in Fig. 1, temporal event graph is a sequence of event graph in ascending time order. Each event graph with a timestamp is composed of multiple co-occurring events of different types, including event actors (as nodes) and event types (as edges). For instances, an event that occurred on \( t - 1 \), was Iran reject British. Identifying knowledge of temporal event graph, such as event actors and their relations, can provide historical clues for predicting multiple co-occurring events of different types at the future timestamp \( t \). In addition, event content also contains some important features (e.g., background information), such as Gulf, oil tanker etc. Incorporating such information can enhance the semantic expression of temporal event graph for better prediction.

However, realizing such multi-event prediction problems in the real world faces many challenges:

- **C1**: Unstructured event content can enhance the semantic expression of event graphs. How to achieve heterogeneous data fusion is a challenging issue.

- **C2**: Multiple co-occurring events imply the structural dependence. How to adaptively model the local neighborhood information of event graphs to capture structural dependence remains a challenge.

- **C3**: Event types among actors in a temporal event graph change significantly over time. How to model the temporal dependence between temporally adjacent events with different types is also a key issue.

To address the aforementioned challenges, we proposed a novel Knowledge-aware attention based temporal Graph Convolutional Network (KatGCN) to predict multiple co-occurring events with different types in the future timestamp. Specifically, we model the social events from open source media as temporal event graph, and extract the background information and topic keywords from event content to enhance semantic. To capture the local structural dependence...
of multiple co-occurring events, we design the knowledge-aware attention based graph aggregation method. Finally, we leverage the long-short term memory network to encode temporal dependence over temporal event graph for multi-event prediction.

Our contributions are summarized as follows:

- We design a novel knowledge-aware attention based graph aggregation method to capture the structural dependence of multiple co-occurring events.
- We develop a new model KatGCN for multi-event prediction, which integrates event content, structural dependences of event graphs and temporal dependence.
- We conduct extensive experiments on five-country datasets to verify the effectiveness of KatGCN and demonstrate the interpretability through a case study.

II. RELATED WORK

Our work is closely related to many literatures on events prediction and knowledge graph learning.

A. Spatio-Temporal Event Prediction

Most existing machine learning methods for event prediction are only suitable for Euclidean or grid like data. For example, a linear regression model [4] utilized tweet frequency to predict the occurrence time of future events. Zhao et al [5] designed a new predictive model based on topic model that jointly characterizes temporal evolution in terms of both the semantics and geographical burstiness. Besides, more complex models, such as multi-task multi-class deep learning model (e.g., SIMDA [6], MITOR [1]), was proposed to predict the subtypes of future events and the scale of spatial events. Recently, Graph Convolutional Network (GCN [7]) has been proposed to address non-Euclidean data in many domains, such as social networks. For instance, DynamicGCN [2] was proposed to encode temporal text features into graphs for forecasting societal events and identifying their context graphs. Besides, REGNN [8] was proposed to learn the impact of historical actions and the surrounding environment on the current events for real-time event prediction.

B. Knowledge Graph Representation

Knowledge graphs (KG), which store real-world facts, is a form of multi-relation graphs. Since each fact changes over time, temporal knowledge graph (TKG) is generated.

Extensive studies have been done on modeling static, multi-relation graph data. For example, RGCN [9] was proposed to deal with the multi-relation graph directly by extending GCN, but it may face over-parameterization as the number of relations increases. Recently, attention mechanism has been applied to knowledge graph representation due to high efficiency and flexibility in modeling graph data. Wang et al [10] developed a novel model KGAT, which explicitly models the high-order connectivity in KG, propagating the embeddings from a node’s neighbors to refine the node’s embedding. Besides, RGHAT [11] was proposed to effectively utilize the neighborhood information of an entity. But the above methods aim to learn the embeddings of nodes and ignore the embeddings of relations. Therefore, CompGCN [12] was proposed to jointly embed both nodes and relations in a multi-relation graph by leveraging a variety of entity-relation composition operations from knowledge graph embedding techniques, which solves the over-parameterization problem.

There are also attempts to model TKG. RE-NET [13] was designed to predict future interactions. EvolveGCN [14] has been proposed for link prediction to capture the dynamism of the graph sequence through using an RNN to evolve the GCN parameters. In addition, a graph learning framework Glean [3] based on event knowledge graphs was developed to incorporate both relational and word contexts.

III. METHODOLOGY

We provide the technical details of our proposed model KatGCN. Fig. 2 shows an overview of KatGCN. The key objectives are (1) integrating the semantic features of event content into event graphs; (2) utilizing neighborhood information to capture the structural dependences between multiple co-occurring events; (3) encoding temporal dependence over different timestamps for multi-event prediction.

A. Problem Definition

Temporal Event Graph (TE graph). TE graph is built on a sequence of event graphs in ascending time order [3]. Each event graph is a multi-relation directed graph with a timestamp, where entities represent event actors and relations represent event types. Let $E$ be a finite set of entities (nodes) and $R$ be a finite set of relations (edges). An event can be defined as a quadruple $(s, r, o, t)$, where $s, o \in E$ and $r \in R$. We denoted a set of events at time $t$ as $G_t = \{(s, r, o, t)\}$. A TE graph can be presented as $G = \{G_{t-k}, G_{t-k+1}, \ldots, G_t\}$.

Problem Formulation. We transform the task of multi-event prediction into a multi-label classification problem to model the occurrence probability of different events at $t + 1$:

$$\{G_{t-k}, G_{t-k+1}, \ldots, G_t\} \xrightarrow{\text{model}} P(Y_{t+1} | G_{t-k}, \ldots, G_t) \quad (1)$$

Where $Y_{t+1} \in \mathbb{R}^{|R|}$ is a vector of event types.

B. Semantic Enhancement

For challenge 1) of Section I, we introduce the semantic enhancement module. We use the pre-trained model sent2vec [15] to get the initial embedding vector $h_t(\bullet) \in \mathbb{R}^d$ of entities, relations and keywords. However, entities, relations and event content are always closely related. Therefore, we extract background and topic keywords from the event content to enhance the semantic expression of event graphs.

1) Entity Semantic Enhancement: We introduce the entity semantic enhancement to incorporate backgrounds into event graphs. For instance, as shown in fig. 1, an event (Iran Reject British) mentioned that Iran denies claims about seizure of British oil tanker in Gulf. Words such as oil tanker, Gulf, show the event background, which can further enhance the semantic integrity of event graphs to improve prediction results.
For a given entity $e$ in $G_t$, we obtain the top ten relevant background words based on TF-IDF algorithm from all the event content at $t$ to enhance the semantic of $e$, as follows:

$$ h'_{e,t} = \tanh \left( W_f \cdot \left( h_{e,t} ; \sum_{\text{word} \in \text{Top}_t} h_{\text{word}} \right) \right) \in \mathbb{R}^d $$

Where $W_f \in \mathbb{R}^{d \times 2d}$ is a learnable weight matrix, and ; is the concatenation operator. If $e$ has no related words, we use zero vector to represent semantic. Then we can get the new entity embedding vector $h'_{e,t}$ (including $h'_{e,t}$ or $h'_{e,t}$) at $t$.

2) Relation Semantic Enhancement: Event graphs contain many edges, which represent event types. Obviously, events with the same event type have similar topic keywords. For example, protest events usually include such words as demonstrate, strike, disturbance, etc. But Yield events usually contain such words as surrender, ousted, etc. To expand the difference between different event types for better relations embedding, we extract topic words for each event type based on LDA model [16] to enhance the semantic of relation $r$, as follows:

$$ h'_{r,t} = \tanh \left( W_k \cdot \left( h_{r,t} ; \sum_{\text{word} \in L_i} h_{\text{word}} \right) \right) \in \mathbb{R}^d $$

Where $L_i, t = [l_{i,1}, \ldots, l_{i,n}]$ is $i$-th row of topic keywords matrix $L_i$ generated from LDA model at $t$, which represents a set of keywords of $n$ topics for $i$-th event type in $G_t$. Thus, we can get a new relation (edge) embedding vector $h'_{r,t}$ at $t$.

C. Knowledge-Aware Attention based Graph Aggregation

For challenge C2 of Section I, we design a novel knowledge-aware attention based graph aggregation method to fully capture the structural dependence between multiple co-occurring events.

1) Knowledge-Aware Attention: Considering that the event graph is a multi-relation graph, the embedding of relations (edges) cannot be ignored. Motivated by GAT [17], we propose a new knowledge-aware attention mechanism, including entity-aware attention and relation-aware attention, to distinguish the importance of neighboring entities and relations.

Relation-Aware Attention. Considering that different relations have different weights when expressing the same entity, we design relation-aware attention. For entity $s$, the relation-aware score represents the weight of each outgoing relation connected to the entity, defined as:

$$ a_{s,r}^t = \text{Attention} \left( W_1 h_{s,t}^r, W_1 h_{r,t}^r \right) $$

$$ a_{s,r}^t = \frac{\exp \left( \text{LeakyReLU} \left( m^T \cdot a_{s,r}^t \right) \right)}{\sum_{r' \in N_s} \exp \left( \text{LeakyReLU} \left( m^T \cdot a_{s,r'}^t \right) \right)} $$

Where $h_{s,t}^r, h_{r,t}^r \in \mathbb{R}^d$ are the embedding vectors of entity $s$ and relation $r$ at $t$, respectively. $W_1$ and $m$ are training parameters. $N_s$ is a set of relations with $s$ as the subject entity. The relation-aware attention score $a_{s,r}$ represents the weights of outgoing relations $r$ when representing the entity $s$.

Entity-Aware Attention The weights of neighboring entities under the same relation may also be different, which inspires the entity-aware attention. We design entity-aware attention to capture the difference in importance between different entities based on the same relation. We regard the object entities based on the same relation as a group, then we calculate the entity-aware attention score, which is defined as:

$$ b_{o,s}^t = \text{Attention} \left( W_2 a_{s,r}^t, W_2 h_{o,s}^r \right) $$

Fig. 2. System framework of the proposed model KatGCN for multi-event prediction. Input data consists of temporal event graph based on social events; We introduce semantic enhancement module to enrich the semantic information of event graph. Then, we design a knowledge-aware attention based graph aggregation method to capture the structural dependencies between multiple co-occurring events. Finally we feed the sequence of TE graph embedding into LSTM to capture the temporal dependence, and add a multi-layer perceptron (MLP) to predict the probability of co-occurring events at $t + 1$. 

MLP
\[
\beta_{o,s,r}^t = \frac{\exp \left( \text{LeakyReLU} \left( n^T \cdot b_{o,s,r}^t \right) \right)}{\sum_{o_j \in N_{s,r}} \exp \left( \text{LeakyReLU} \left( n^T \cdot b_{o_j,s,r}^t \right) \right)}
\]  

(7)

Where \( h_{o,t}^r \) is the embedding of the entity \( o \) under relation \( r \) and entity \( s \), \( N_{s,r} \) represents a set of object entities of \( s \) under relation \( r \). \( W_2 \) and \( n \) are training parameters. The entity-aware attention score \( \beta_{o,s,r}^t \) shares all the object entities information of the same subject entity under the same relation at \( t \), which is beneficial to capture the association between different co-occurring events under the same relation.

2) Graph Aggregation: The event graph is multi-relation directed graph. We need to get the embedding of entities and relations to get the event graph representation. Inspired by CompGCN [12], we design a novel knowledge-aware attention based CompGCN to learn the the event graph representation.

Specifically, we leverage the entity-relation composition operation [18] based on the knowledge-aware attention to incorporate the embedding of entities and relations into the GCN. For an entity \( s \) in \( \mathcal{G}_t \), we apply the knowledge-aware attention based CompGCN to update its embedding vector:

\[
h_{s,t}^{t,(l+1)} = f \left( \sum_{(r,o) \in \mathcal{E}(s)} W_q(l) \Phi \left( \alpha_{s,r}^{l,t} h_{r,t}^{t,(l)}, \beta_{o,s,r}^{l,t} h_{o,t}^{t,(l)} \right) \right)
\]

(8)

Here, \( \Phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d \) is a composition operator. We choose multiplication as \( \Phi \). \( h_{r,t}^{t,(l)} \) and \( h_{o,t}^{t,(l)} \) denote feature embedding in \( l \)-th aggregation layer for relation \( r \) and entity \( o \), respectively. \( W_q \) is a relation-specific parameter. \( f(\bullet) \) is the ReLU activation function. Next, we update the embedding vector of relation \( r \) in \( \mathcal{G}_t \):

\[
h_{r,t}^{t,(l+1)} = W_r(l) h_{r,t}^{t,(l)}
\]

(9)

Where, \( W_r(l) \) is a learnable transformation matrix in the \( l \)-th layer, which can project all the relations to the same embedding space as entities, so that the prediction task can perform operations on the nodes and edges uniformly.

To summarize, the advantage of our graph aggregation lies in distinguishing the importance of different neighboring entities and relations. We apply two layers to realize the aggregation of two-hop neighborhoods. For \( \mathcal{G}_t \), we obtain the embedding matrix \( H_t^r \) of entities and \( H_t^o \) of relations.

D. Event Prediction

1) Temporal Encoding: For challenge C3 of Section I, we utilize a temporal encoding module to capture temporal dependence between temporally adjacent events. Given a sequence of embedding matrix of entities and relations, i.e., \( \{ H_{t-k:t}, H_{t-k+1:t} \} \), we apply LSTM to encode historical information in the TE graph, aiming to model temporal dependence from the graph sequence. To reduce the spatial of feature embeddings and obtain salient feature, we employ the max pooling operation over the embedding matrix of entities and relations, respectively. Then, we feed them into the LSTM model to get the historical global embedding \( X_t \):

\[
X_t = \text{LSTM} \left( \left[ p(H_t^e); p(H_t^r) \right], X_{t-1} \right)
\]

(10)

Where, \( p(\bullet) \) represents the max pooling operation applied element-wise over all nodes or edges.

2) Multi-event Prediction: Through temporal encoding, we have obtained the historical embedding \( X_t \) up to time \( t \). Then, we model the probability of multiple co-occurring events in the future timestamp \( t+1 \) based on TE graph:

\[
P \left( Y_{t+1} \mid \mathcal{G}_{t-k,t}, \ldots, \mathcal{G}_t \right) = \sigma \left( W_{p} X_t \right)
\]

(11)

We feed the \( X_t \) into a MLP to calculate the probability of different event types. We define the MLP as a linear softmax classifier parameterized by \( W_p \). \( \sigma \) is a nonlinear function.

Next, we adopt the categorical cross-entropy [19] loss:

\[
\mathcal{L} = - \frac{1}{|R|} \sum_{i \in R} y_i \ln \left( \frac{\exp \left( \hat{y}_i \right)}{\sum_{j \in R} \exp \left( \hat{y}_j \right)} \right)
\]

(12)

Where \( \hat{y}_i \) is the model prediction for event type \( i \) before the nonlinear function (\( \sigma \)) in (11).

IV. EXPERIMENTS AND RESULTS

We evaluate the performance of KatGCN for multi-event prediction. We aim to answer the following key questions: (1) Whether KatGCN achieve satisfactory predicting results compared with other baselines; (2) Whether different modules in KatGCN can improve the experimental results better; (3) Whether the results of KatGCN have better interpretability.

A. Datasets and Evaluation Metrics

The experimental evaluation was conducted on the Global Database of Events, Language, and Tone event data (GDELT1). It contains political events designed to assess national and international crisis events. These events are divided into 20 main types and 220 subtypes such as Appeal, Yield, Protest etc. Each event is coded into 58 fields including date, actor attributes (actor1, actor2), event type, source (event URL) etc. In this paper, we focus on all subtypes of events and select country-level datasets from five countries (Iran, Iraq, Saudi Arabia, Syria, and Turkey) from January 1, 2018 to June 20, 2020. We split the dataset of each country into three subsets, i.e., train(80%), valid(10%), test(10%). The time granularity is one day. We use the three metrics to evaluate the results of the experiment, including F1-score, F2-score and Recall.

B. Comparative Methods

We compare KatGCN with some state-of-the-art baselines:

- **DNN**: We feed TF-IDF text features to a deep neural network for events prediction.
- **RE-NET [13]**: It contains a recurrent event encoder and a neighborhood aggregator to infer future facts.
- **Glean [3]**: This is a temporal graph learning method with heterogeneous data fusion for predicting multi-event.

Next, we conduct ablation studies:

1https://www.gdeltproject.org/
Table I: Prediction results of KatGCN and baselines over all datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Iran</th>
<th>Iraq</th>
<th>Saudi Arabia</th>
<th>Syria</th>
<th>Turkey</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F1</td>
<td>F2</td>
<td>Recall</td>
<td>F1</td>
<td>F2</td>
</tr>
<tr>
<td>DNN</td>
<td>49.08</td>
<td>54.71</td>
<td>59.62</td>
<td>53.07</td>
<td>58.44</td>
</tr>
<tr>
<td>RE-NET</td>
<td>56.20</td>
<td>60.21</td>
<td>62.99</td>
<td>55.46</td>
<td>62.04</td>
</tr>
<tr>
<td>Glean</td>
<td>57.20</td>
<td>64.88</td>
<td>73.06</td>
<td>59.05</td>
<td>70.25</td>
</tr>
<tr>
<td>KatGCN-semantic</td>
<td>66.15</td>
<td>77.04</td>
<td>79.62</td>
<td>66.04</td>
<td>72.05</td>
</tr>
<tr>
<td>KatGCN-attention</td>
<td>60.54</td>
<td>68.02</td>
<td>71.93</td>
<td>64.09</td>
<td>69.03</td>
</tr>
<tr>
<td>KatGCN</td>
<td>68.33</td>
<td>78.83</td>
<td>79.95</td>
<td>67.34</td>
<td>72.37</td>
</tr>
</tbody>
</table>

C. Experiments Results

We evaluate the prediction performance of our proposed model across five datasets. To avoid errors caused by randomness, we obtain an average of 10 experiments on each dataset. Table I presents comparison and ablation results.

1) Prediction Performance: Our model KatGCN outperforms all other baselines on five datasets. From the above comparison results, we have the following observations:

- KatGCN-semantic: without the semantic enhancement module. We only consider the structure-graph information and temporal dependency in the TE graph.
- KatGCN-attention: without the knowledge-aware attention module. We only use the classical CompGCN to achieve event graph aggregation.

- KatGCN presents the best performance on the five datasets. The reason is that we introduce a knowledge-aware attention mechanism to make full use of the neighborhood information of the event graph.
- Graph based methods (RE-NET, Glean, KatGCN) are obviously better than static features based methods (DNN), which shows that the graphs can model structural dependence between different events (e.g., sharing actors).

2) Ablation Experiments: From the results of ablation experiments, we can observe the following findings:

- The semantic enhancement is also essential, which can slightly improve the performance of multi-event prediction by enriching the semantics of event graphs.

D. Sensitivity Analysis

We study the parameter sensitivity analysis of KatGCN, mainly including: the embedding dimensions (d), layers of graph aggregation (l), and the time step of history (k):

1) Embedding Dimensions: We study how the embedding dimensions affect the model performance. As shown in Fig. 3(a), the performance improves obviously with d increases when d is below 100. Higher d will not bring significant performance improvement and may cost more training time.

2) Layers of graph aggregation: The number of layers l represents the hops of neighbors that nodes aggregate. Fig. 3(b) shows the impact of different l. Compared with 1-layer, 2-layer significantly improves the performance. But the performance is almost unchanged when l increases. We infer that there is overfitting due to the increase in parameters.

3) Time Step of History: We need to encode events information of past k time step. Fig. 3(c) illustrates the performance with different k. The performance reaches the best when k is 7. But larger k is not likely to go higher performance.

E. Case Study

We present a case to show how the proposed model identifies historical event information to predict multi-event in the future. Then we verify the interpretability of knowledge-aware attention based graph aggregation method via an example.

1) Identify historical events: We select a series of social events from the Iran datasets as a case. We utilize the TE graph of past week to successfully predict multiple events on January 16, 2020. As shown in Fig. 4, we describe a series of social events about the shooting down of Ukraine International Airlines Flight. We find that student initiated an event of demonstrate or rally on January 11. Then, government criticize or denounce Ukrain on January 12 and citizen try to express intent or negotiate to Iran and threaten the Tehran on January.
13 and 14 respectively. The events that occurred between different actors and temporal dependence were successfully captured by our model. In the prediction result, our model correctly predicted the possible events on January 16.

2) Interpretability: Benefiting from the knowledge-aware attention, we show the interpretability of our model. As shown in Fig. 4, We take some co-occurring events on January 13, 2020 as an example. We chose citizen as the central actor, and calculated different attention scores of neighboring relations and entities during aggregation. We observe that the event of Fight with small arms and light weapons has larger attention score. Besides, for the event of Express intent to meet or negotiate, the entity Ukarain has a larger attention score than Iran. This is the result we expected, and also consistent with the historical events and our prediction result.

V. CONCLUSION

In the paper, we propose a Knowledge-aware attention based temporal Graph Convolutional Network (KatGCN) for multi-event prediction. Specifically, we first model social events as TE graph and extract event background and topic keywords from event content to enhance semantic expression of event graphs. Then, we design a knowledge-aware attention based graph aggregation method to fully use the neighborhood information and capture structural dependency between co-occurring events. Finally, we utilize temporal encoding to capture temporal dependence between temporally adjacent events. Experiments on five-country datasets show KatGCN significantly outperforms the state-of-the-art baselines and has interpretability. Future work will consider predict event actors of different events to infer a complete social event.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China under Grants No.2018YFC0831703 and Postgraduate Scientific Research Innovation Project of Hunan Province CX20200015.

REFERENCES


SolDetector: Detect Defects Based on Knowledge Graph of Solidity Smart Contract

Tianyuan Hu, Zhenyu Pan, Bixin Li
School of Computer Science and Engineering
Southeast University, Nanjing, China

Abstract—Smart contract security is one of core security issues in the application of blockchain. In recent years, attacks on smart contracts occur frequently, there are a lot of researches concerning on smart contract security issues. However, almost all solutions proposed in these researches are low precision and high False Negative Rate (FNR). In this paper, we propose a defect detection method for checking security of Solidity smart contract based on knowledge graph. Therefore, we first construct knowledge graph of smart contracts by fully integrating syntax and semantic information of Solidity source code; then, we define defect patterns by analyzing defect characteristics; furthermore, we define inference rules for defects based on knowledge graph and defect patterns; finally, we detect defects by SPARQL query. We also implement a tool named SolDetector and perform experiment on three different datasets, which shows that SolDetector is effective and efficient.

Index Terms—smart contract, knowledge graph, defect detection

I. INTRODUCTION

As a distributed public ledger technology in peer-to-peer network, blockchain provides an innovative method to store information, execute transactions, and build trust in an untrusted environment. Even though Blockchain technology provides many new mechanisms to solve security issues compared to traditional technologies. However, blockchain security is a bottleneck affecting its wide use because of the existence of vulnerabilities in smart contracts, consensus protocols, infrastructure code, etc.

Ethereum, as a representative of public chain, uses a high-level programming language called Solidity to write its smart contract. In Ethereum, a smart contract is actually a collection of codes, including various functions and various states generated by the code running process. It is compiled to Ethereum Virtual Machine (EVM) instructions for blockchain deployment. Once published on Ethereum platform, smart contracts will be executed on all nodes in the network as a program, and cannot be modified. If the deployed smart contract’s code is insecure, software vulnerabilities may be exploited by malicious attackers. As smart contract code involves digital assets, it may cause huge losses once the defects of the contract code are used. Thus, how to ensure the security of the smart contract is very important.

Over the past few years, the automated analysis tools for smart contracts have made progress. Mainstream defect detection methods can be divided into static analysis [1]–[3] and dynamic analysis [4], [5]. Static analysis focuses on syntax analysis of source code, which is not suitable for complex logic analysis. For some defects with complex logic, static analysis has a low precision. Dynamic analysis has a high precision because it detects real smart contract vulnerabilities during contract executions. But dynamic analysis fails to achieve sufficient code coverage, which ignores some syntax errors and produces false negatives. So there are two open challenges in detecting smart contract.

How to increase the precision of the contract defect detection method and keep low false negatives?

How to detect more defect types of smart contract and extend the method flexibly?

In this paper, we propose a defect detection method for smart contract in Solidity based on knowledge graph to improve the precision and find more defects. We summarize our contributions as follows:

- The knowledge graph of smart contract is constructed, including the ontology layer and the instance layer.
- A defect detection method is proposed for checking security of Solidity contract based on knowledge graph, which realizes defect localization by inference rules and SPARQL.
- A tool called SolDetector is implemented to fully automate the analysis of contracts.
- An evaluation is performed to demonstrate the effectiveness and efficiency of SolDetector over three different datasets including 24,583 smart contracts.

The rest of the paper is organized as follows. Section II introduces the background knowledge of smart contract defect and knowledge graph. Section III discusses the knowledge graph construction of smart contract. Section IV details defect detection of smart contract based on knowledge graph. Section V evaluates SolDetector by experiment. Section VI discusses related work. Finally, Section VII concludes the paper and suggests future work.

II. BACKGROUND KNOWLEDGE

A. Smart contract defect

A contract defect is an error, flaw or fault in a smart contract that causes it to produce an incorrect or unexpected result, or to behave in unintended ways [6]. Smart contract defects are mainly caused by coding and may be related to
developers, implementation language, compilers, and execution mechanism of blockchain system. We collected smart contract researches and 16 kinds of contract defects with characteristics are concluded, including Missing Reminding Execution Results defect, Balance Manipulation defect, Integer Overflow and Underflow defect, etc.

B. Defect pattern

The defect pattern is an abstract representation for defects capturing defect characteristics, including core elements, relationships between elements and restrictions on elements. A reasonable definition of defect pattern implies satisfaction of the contract defect. For example, a defect pattern of Reentrancy defect is defined as below.

1. ReentrancyPattern ≡
2. ∃ containAssignment. Assignment ⊓
3. ∃ callFunction. FunctionCall ⊓
4. (Assignment, follow, FunctionCall) ⊓
5. (FunctionCall, without, gasLimitation)

In the definition, a pattern is composed of core elements, relationships and other restrictions. Core elements in Reentrancy defect pattern are Assignment(line 2) and FunctionCall(line 3). The relationship between Assignment(line 2) and FunctionCall is follow(line 4). There is also a gasLimitation on FunctionCall(line 5). A core element involved in the defect pattern can be described as a class in knowledge graph. Similarly, a relationship can be described as an object property between two classes, which restricts the logical relationship between elements. Specific defects not only have logical relationship restrictions between elements, but also have limitation on elements themselves(line 5).

C. Knowledge graph

Knowledge graph is a technical to describe knowledge and construct connections between all things in the world using graph models [7]. It consists of nodes and edges. Nodes are individuals or abstract concepts. Edges are properties of individuals or relationships between individuals. Based on knowledge graph, we can identify, discover and infer complex relationships between things and concepts from data. Knowledge inference is the process of inferring unknown facts or relations based on existing facts or relations in the graph and applying certain rules to draw logical conclusions. The knowledge graph of smart contracts represents the basic syntax and semantic of smart contracts. Furthermore, more complex unknown facts that can be obtained by inferring.

III. KNOWLEDGE GRAPH CONSTRUCTION OF SMART CONTRACT

Fig.1 depicts knowledge graph construction process. Knowledge graph is constructed based on the source code of a smart contract. Combined with Solidity grammar, Abstract Syntax Tree (AST) is generated to extract information for building knowledge graph. The knowledge graph of smart contract integrates two layers: ontology layer describing abstract concepts and instance layer describing concrete facts.

A. Ontology layer

Ontologies are artifacts used to model and represent knowledge related to a specific domain in an explicit way [8]. A typical ontology consists of a finite number of terms and relations between them. Terms are important concepts of the given domain. The smart contract focused in the paper is written in Solidity. Therefore, how to model and represent the Solidity by ontology is illustrated in this section.

The purpose of ontology layer is to represent Solidity in terms of concepts and relations. In this paper, the ontology layer describes code elements and corresponding relationship of Solidity source code. Code elements are modeled as classes and relationships between code elements are modeled as object properties.

For example, a Solidity code snippet is shown in Fig.2. The contract has four main code elements, including a state variable, a function, a function call and an assignment. Four main classes can be abstracted from the code snippet, including Contract, Function, StateVar, FunctionCall and Assignment. Correspondingly, relationships between classes can be concluded, such as hasStateVar, containAssignment. Hence, object properties can be abstracted as hasFunction, containAssignment and hasStateVar.

The illustration of the class and object property for the code snippet1 can only represent part of key elements. Full knowledge graph definition of smart contract contains 13 types of classes and 26 types of object properties.

B. Instance layer

As the definition of classes and object properties in ontology layer, the next step is to extract required information from source code to build the instance layer and construct the entire knowledge graph. In Fig. 1, the information extractor extracts key facts as individuals and attaches relationships between individuals based on AST of source code. An abstract syntax tree parser ANTLR generates AST of smart contract source.
code. Information in each AST node can be accessed by visitor. The tool can be extended to support other smart contract language by adding an ANTLR grammar. To deal with the instance extraction for multiple classes or object properties, the node visitor is defined for each class. A node visitor is responsible for the individuals generation of one class. Details of extracting individual and relationship are illustrated below with the explanation for the code snippet1.

Extracting individual. Class is an abstract concept used to describe key code elements in smart contract. Each component in source code can be extracted as an individual belonging to a class. Given the class Function in the code snippet1, it’s visitor enumerates all Function and generates an individual of Function with name “withdraw”.

Extracting relationship. Relationship between individuals can be extracted according to the object property definition. Relationship between different individuals can be extracted during the nested visits by visitor, which is called syntax relationships in this paper. For instance, when the visitor from the Contract node to the Function node, hasFunction property associates Function individual with it’s declaring Contract SimpleWithdraw.

In addition, relationships include not only syntax relationship defined in smart contract, but also logical relationship. For the code snippet1, the logical relationship is currently reflected in order of execution statement within a function, as reflected by follow, e.g. (Assignment, follow, FunctionCall) (Line4 and Line3). Finally, the knowledge graph constructed for the code snippet1 is shown in the Fig. 3. More specific description are omitted for brevity.

We use OWL as an ontology description language, which is a rich vocabulary description language that can characterize relationships between classes, types of properties and characteristics of properties. The proposed method refers to OWL documents as a knowledge graph. In OWL, line numbers are stored in the individual name and help to localize defects in source code.

IV. Defect detection of smart contract

Based on knowledge graph of smart contract, we conduct defect detection process as shown in Fig.4. The process consists of two main steps: knowledge inference and defect localization.

A. Knowledge inference

In this step, indirect knowledge is inferred upon the knowledge graph which contains facts extracted from the source code. The inference process infers the indirect relationships between code elements from known facts and relationships. For example, direct logical relationship of two adjacent code elements represented by “follow” in instance layer. Indirect logical relationship can be inferred from basic relationships. The facts can be expressed as triples. By matching triples, the inference rule for logical relationship can be expressed as:

\[
(?A \text{ follow} ?B), (?B \text{ follow} ?C) \Rightarrow (?A \text{ follow} ?C)
\]

In the rule, the variables start with a question mark (?) denote matched subjects or objects of triples [9]. The indirect logical relationship between A and C can be inferred. Similarly, the inference rule for judging types of s state variable in and arithmetic operations can be expressed below, which is critical for identifying Integer Overflows and Underflow defect.

\[
(?x \text{ is Assignment}), (?x \text{ assignObject} ?y),

(?y \text{ typeName} ?type), (?\text{type} \text{ like} \text{ ’uint’})
\]

Inference rules are applied to capture complex relationships and abstract elements satisfying the definition of the defect pattern. Even basic fact from AST are not sufficient, complex relationships and code details critical to defect detection can be drawn by inference rules. An example that recognizes Reentrancy defect by inferring is explained below. In Fig.5, the inference rule of Reentrancy defect is shown. In contract SimpleReentrancy, Function\_withdraw calls a function (1) realizing transferring by “call” and contains an assignment (3) subtracting the transferring amount from credit[msg.sender]. According to Reentrancy defect pattern, the contract contains main code elements of Assignment and FunctionCall. Besides, FunctionCall\_require realizes transferring by “call”, which has no gas limitation (2). Combined with the logical relationships inference rule, it is inferred that Assignment\_credit[msg.sender] follows FunctionCall\_require (4). Therefore, FunctionCall\_require and Assignment\_credit[msg.sender] may lead a malicious contract calls back into the contract before the first invocation of the function is finished. Because SimpleReentrancy contains all code elements in Reentrancy defect and meet other restrictions, the contract has Reentrancy defect.

In this section, only one example is given. For other contract defects, inference rules are customized according to defect
characteristics. Furthermore, inference rules are universal for smart contract in different program language.

B. Defect localization

The inference rules use SPARQL language to query and manipulate the knowledge graph data. Based on all available definition in ontology layer and instance layer, the SPARQL of a defect is able to utilize main elements and restrictions. From the perspective of defect matching, SPARQL enables flexible search strategies based on knowledge expressed in higher levels. Based on triple matching, SPARQL query is conducted on knowledge graph. Successful triple matching will return an individual name, which shows a defect is found; otherwise, no defect is found. Due to the line number is stored in each individual name, there are following two cases of successful SPARQL query.

1. Case1: A defect pattern is satisfied (e.g., Integer Overflow and Underflow defect) and an individual name containing specific location is returned, which indicates the defective statement.
2. Case2: A defect pattern is satisfied (e.g., Frozen Ether defect) and the contract individual name is returned. Thus, the contract has this defect and cannot be localized to specific line.

For example, one challenge in identifying Integer Overflows and Underflow defect is judging the type of an operation variable in an assignment. We illustrate the SPARQL query of Integer Overflow and Underflow defect as shown in Fig.6.

V. EVALUATION

To put the proposed method into practice, we have implemented SolDetector based on Jena. It integrates the Antlr generating AST and information extractor building knowledge graph for smart contract dynamically. The defect inference rules are executed by SPARQL query engine on OWL files.

We use three datasets to evaluate SolDetector. Dataset1 consists of 179 smart contracts selected from test datasets of three popular analysis tools [1], [2], [10] and has been attacked in a real-world, totaling 31,904 lines of the code. Dataset2 consists of 15,623 smart contracts crawled from Etherscan in 2018, totaling 4,197,965 lines of the code. Dataset3 consists of 8,781 smart contracts crawled from Etherscan in 2020, totaling 5,215,734 lines of the code. There is no restriction on Solidity contract versions and contracts’ lines. It is useful to estimate the efficiency of SolDetector for massive contracts with different Solidity version.

A. Effectiveness of SolDetector

To evaluate the effectiveness of defect detection, we run SolDetector on three datasets and use measurements listed below. 1) TP indicates the number of vulnerable contracts detected by the tool correctly. 2) FP indicates the number of vulnerable contracts detected by the tool incorrectly. 3) TN indicates the number of contracts without defects detected by the tool correctly. 4) FN indicates the number of contracts free of defect detected by the tool incorrectly. Precision, Recall, FDR and FNR can be calculated as: Precision = TP / (TP + FP) × 100%, Recall = TP / (TP + FN) × 100%, FDR = FP / (TP + FP) × 100%, FNR = FN / (TP + FN) × 100%.

Dataset1 contains famous vulnerable contracts with distinct defect characteristics. While Dataset2 and Dataset3 consist of contracts randomly crawled. In order to simplify the evaluation, the careful analysis is mainly aimed at Dataset1 and 10% contracts of Dataset2 and Dataset3 were randomly selected to calculate TP, TN, FP, FN. Evaluation of SolDetector is shown in Table I.

For Dataset1, SolDetector successfully detected 125 real vulnerable contracts. Meanwhile, it mistakes 4 safe contracts as vulnerable. The number of vulnerable contracts omitted by SolDetector is 12. We manually analyze 12 vulnerable contracts omitted by the tool. There are 7 contracts containing Reentrancy, 2 contracts containing Balance Manipulation, 1
The following result can be drawn: 1) for seven defects, Recall of SolDetector is greater than or equal to SmartCheck. 2) for eight defects, FNR of SolDetector is less than or equal to SmartCheck. 3) for only one defect (D7), SolDetector has a lower Recall of 81.82%. In general, SolDetector has a higher Recall and lower FNR, which shows the advantage of our method in detecting defects.

Furthermore, we also compare SolDetector with other tools to demonstrate its efficiency. Tool’s efficiency is related to the defect number that can be detected and the contract size. Efficiency information are collected from corresponding papers. SolDetector can detect more defects than Securify [10] and SmartShield [13]. Moreover, Securify and SmartShield cost 30s and 28s per contract, which shows that both of them are inefficient. SmartCheck is suitable for more defects and larger contract, whose average time is 1.66s per contract that is longer than SolDetector’s average time. Thus, SolDetector is the fastest tool, followed by SmartCheck, Securify and SmartShield.
VI. RELATED WORK

As a distributed public ledger technology in peer-to-peer networks, blockchain is increasingly used in various fields. However, there are still security issues in smart contracts, which affects further promotion of blockchain technology. It is necessary to fully analyze potential security threats to avoid defects as much as possible. At present, the existing smart contract defect detection methods focus on symbolic execution, model checking, fuzzy testing and other methods.


SolDetector makes up the shortcomings of static analysis and dynamic analysis. Knowledge graph construction for smart contract can be easily modified according to detection needs for any new code elements. Even though Solidity grammar updates and new code element are added, it is easier to extract information by generating AST with our customized Solidity grammar. Moreover, new defect patterns and corresponding inference rules can be flexibly expanded. A detection method suitable for more known defects and can be extended flexibly for new defects is significant.

VII. CONCLUSION

In this paper, we propose SolDetector, a tool for smart contract defect detection. SolDetector fully integrates syntax and semantic information of smart contracts to construct knowledge graphs by a personalized pluggable information extractors. Smart contract will be scanned for 16 kinds defect by inference rules and corresponding SPARQL, which realizes the defect localization efficiently.

Our method cannot analyze the contract execution state, which leads to a high FNR for Reentrancy defect. Combining static with dynamic analysis might be a potential way to address the disadvantage. The method proposed in this paper provides a sound basic for the combination of static and dynamic analysis. In future work, we will track and analyze the execution information to enrich the knowledge graph of smart contracts.

REFERENCES

SMART: Towards Automated Mapping between Data Specifications

S. Kalwar*, M. Sadeghi*, A. Javadian Sabet*, A. Nemirovskiy* and M. Rossi†

* Dipartimento di Elettronica, Informazione e Bioingegneria
† Dipartimento di Meccanica
Politecnico di Milano
Piazza Leonardo da Vinci 32, I-20133 Milano, Italy
Email: firstname.lastname@polimi.it

Abstract—The ability to perform automated conversions between data conforming to different specifications is a key ingredient to achieve interoperability among heterogeneous systems—which, in turn, is at the basis of the creation of so-called Systems of Systems. These conversions require the definition of mappings between concepts of separate data specifications, which is typically a hard and time-consuming task. In this paper, we present a technique to automatically suggest mappings to users, based on both linguistic and structural similarities between terms. The approach has been implemented in our prototype tool, SMART (SPRINT Mapping & Annotation Recommendation Tool), and it has been validated through tests carried out using specifications from the transportation domain.

Keywords—Ontology, linguistic similarity, structural similarity, automated mapping, natural language processing

I. INTRODUCTION

In recent years there has been a growing interest in the development and deployment of so-called Systems of Systems [1], [2], where independent, heterogeneous systems built using different technologies interact to provide complex services. A paradigmatic example of this trend is found in the transportation domain, especially in the European Union, where initiatives are underway to create a Single European Transportation Area [3], and in particular a Single European Railway Area [4]. Most prominently, the EU Shift2Rail Joint Undertaking [5], especially within its Innovation Programme 4, aims to provide users with a “one-stop-shop” solution that allows them to handle multi-modal trips across borders, using a single application that integrates many different services (shopping, booking, ticket issuing, etc.) from heterogeneous providers of different Nations. This requires the integration of services offered by transport operators from different countries, which typically use different standards and specifications to describe data such as travel offers, booking information, etc. This heterogeneity of data representations significantly hinders the interoperability of the systems to be integrated, and it can be mitigated through the adoption of suitable conversion mechanisms between data specifications. Scrocca et al. [6] developed a promising data conversion approach, following the schema described in [7] and shown in Figure 1.

In this schema, a reference ontology acts as “pivot” between data specifications A and B, whereby specification A is “lifted” to the ontology (i.e., the concepts in specification A are mapped to those in the ontology), and then the latter is “lowered” to specification B. The approach has proven to be effective [6] and, although it originated from projects focusing on the transportation domain and has been tested using transportation data, it is general and can be applied to any other domain where a reference ontology is available.

At its core, the schema of Fig. 1 relies on declarations—expressed in suitable notations—that precisely establish correspondences between terms of specifications A, B with concepts in the reference ontology. The creation of these mappings, however, is typically a time-consuming activity, which must be carried out by users who have a good level of familiarity with the data specifications and with the reference ontology.

This paper presents a technique that aims at easing the process of creating mappings between concepts in different data specifications and ontologies by: (i) suggesting to users potential mappings between the terms in a specification and those of an ontology; (ii) allowing them to review and confirm/review the suggested mappings; and (iii) automatically generating the necessary annotations and declarations that enable the conversion technique of [6].

![Fig. 1. General schema of conversion mechanism (from [6], [7]).](image)

The technique builds on the principles laid out in [8]. It assumes that data specifications are provided as XSD files, and it is based on two main steps. First, it looks for linguistic similarities between the terms of a given data specification and those in the reference ontology, thus creating an initial mapping. Then, it uses the structure of the specification and
of the ontology (i.e., how terms are related to one another) to refine—and possibly extend—the linguistic mapping.

The technique has been implemented in a prototype tool, called SMART (SPRINT Mapping & Annotation Recommendation Tool), as part of the SPRINT [9] project, which aims to define an innovative Interoperability Framework [10]. The tool supports users in the creation and review of the mappings, and then in the generation of the corresponding annotations. The technique and the tool have been validated through a set of mapping experiments involving data specifications and ontologies from the transportation domain.

The paper is structured as follows: Sect. II overviews some relevant related works; Sect. III describes the procedures for generating the suggested mappings and the corresponding annotations; Sect. IV briefly describes the SMART tool; Sect. V presents the results of the validation, and Sect. VI concludes.

II. RELATED WORK

In the domain of mappings between XML-based data and ontologies, most works focus on automatically transforming XML Schemas into newly-created ontologies capturing the implicit semantics existing in the structure of XML documents. For example, Rodrigues et al. [11] specify mappings between the elements of an XML Schema and those (classes, object and datatype properties) defined by an OWL ontology. OWL elements are identified by their URI references [12], while the mapped XML nodes are identified by XPath [13] expressions.

When transforming XML-based information into an ontology, two approaches are most common: in the first approach, mapping rules between elements of the XSD and OWL standards are used to generate an ontology from an XSD file; in the second approach, instead, the generated ontology is populated from XML instances. Hacherouf et al. [14] focus on the first approach. They use a set of transformation patterns based on the Janus method [15] to translate an XSD block to an equivalent ontology element. The Janus method uses a greater number of XSD elements [16] compared to the work in [11], where transformation rules are limited to the most-used XSD elements (xsd:element, xsd:attribute, xsd:complexType). Some works follow a linguistic approach to translate XML-based information into ontologies. Among them, An et al. [17] propose a heuristic algorithm for finding semantic mappings based on tree pattern formulas [18]. Yin et al. [19] define a method to create mappings between the concepts of two different ontologies. The method divides each ontology into several sub-trees using a classification method [20], and builds mappings between the root nodes of the identified trees in the ontologies. Word similarity is defined based on the assumption that the longer the common substrings between two terms, the more similar they are, and it is computed using the Longest Common Substring algorithm [21]. Shen et al. [20] present a method to compute contextual similarity between two words. The idea is that two concepts can be mapped when they either have high word similarity and low context dissimilarity, or low word similarity and high context similarity.

Our proposed technique is unique in that it employs a two-step process that combines both a linguistic and a structural approach to map elements between XSD specifications and ontologies. This has the advantage that, even when schema elements do not correspond structurally, they might still be linguistically similar, which makes it possible to establish suitable correspondences. Some works (e.g., the Janus method [15]) cover a greater range of XSD features than our approach when transforming XSD schemas into OWL ontologies. However, on the one hand, we pursue a different aim, in that we do not generate new ontologies, but identify correspondences between existing elements; on the other hand, our approach exploits both linguistic and structural features, and we leave for future work the extension of the breadth of XSD features taken into account by the algorithms.

The next section provides some details about the proposed technique.

III. METHODOLOGY

The overall workflow of the approach implemented in the SMART tool is depicted in Fig. 2. Given a pair of specifications, SMART identifies a set of mapping suggestions, where each suggestion is a pair of terms—one from each specification—accompanied by a Confidence Score (CS). Then, the selector module receives the mapping suggestions and allows the user to manually inspect them; during the inspection, the user can confirm or modify the mappings, and even suggest new pairs, if necessary. Alternatively, the user can let the SMART tool automatically choose the suggestions with the highest CS. Finally, the pipeline sends the Confirmed Mappings to the Annotation Generation module to produce the annotations.

Fig. 2. Overview of the workflow implemented in the SMART tool.
The rest of this section describes the procedure for generating the pairs of suggested mappings, whereas Section IV provides an overview of the implementation of the tool.

A. Mapping Algorithm

Given two different data specifications, the Mapping Algorithm’s primary idea is to identify linguistically and structurally similar terms. The algorithm uses two main techniques: (i) linguistic mapping and, then, (ii) structural mapping. The former applies Natural Language Processing (NLP) and Machine Learning (ML) techniques to identify similar terms. The latter, instead, exploits the source and target data specifications’ structures to refine—and possibly extend—the set of mappings. Algorithm 1 details the flow of the Mapping Algorithm. For clarity and brevity, some parts have been encapsulated in sub-algorithms that are shown as algorithms 2, 3 and 4.

The algorithm takes as input two specifications. Typically, one of them is represented by an XSD file (X in line 2 of Alg. 1), whereas the other is an ontology represented by an OWL file (O). However, the algorithm can also work when the input files are both XSD files, or both ontologies.

Algorithm 1 Mapping Algorithm

1: procedure SMARTMAPPING
2: input: X: XSD file, O: OWL file
3: output: P: set of triples (xt, ot, s), xt ∈ X, ot ∈ O, s:Confidence score
4: xName ← (X.Attribute.name ∪ X.Element.name)
5: xType ← (X.Attribute.type ∪ X.Element.type)
6: xCl ← X.ComplexTypes
7: xObPr ← \{x|∀xb ∈ xName if XType(xName) = ComplexType\}
8: xDtPr ← \{x|∀x ∈ xName if XType(xName) = Datatype\}
9: oCl ← O.Class
10: oObPr ← O.ObjectProperty
11: oDtPr ← O.DatatypeProperty
12: procedure
13: ▷ Create initial mappings between terms using linguistic similarity
14: mappedClass ← W2VlinguisticMap(xCl, oCl)
15: mappedObjProp ← W2VlinguisticMap(xObPr, oObPr)
16: mappedDataProp ← W2VlinguisticMap(xDtPr, oDtPr)
17:
18: ▷ New mappings between object properties (Alg. 2)
19: mappedObjProp ← AddObjPropFromClasses(
20: mappedObjProp, mappedClass, xObPr, oObPr )
21: procedure
22: ▷ New mappings btw. classes based on classes & obj. prop. (Alg. 3)
23: mappedClass ← AddClassesFromClassesAndObjProp(
24: mappedObjProp, mappedClass,
25: xCl, oCl )
26: procedure
27: ▷ New mappings between classes based only on properties (Alg. 4)
28: mappedClass ← AddClassesFromProp(
29: mappedObjProp, mappedClass,
30: xCl, oCl )
31: procedure
32: return mappedClass \∪ mappedObjProp \∪ mappedDataProp
33: end procedure

Algorithm 2 New Object Properties

1: procedure ADDOBJPROPFROMCLASSES
2: input: mappedObjProp, mappedClass, xObPr, oObPr
3: output: P: set of triples (xt, ot, s), xt ∈ X, ot ∈ O, s:Confidence score
4: procedure
5: propList ← \{\}
6: foreach (xcl, ocj, s) ∈ mappedClass do
7: foreach (xcj, ocj, s) ∈ mappedClass do
8: foreach (xp, op) ∈ xObPr \× oObPr do
9: if xcj = xop, ComplexType & ocj = op, Domain &
10: s ← (s1 + s2)/2
11: propList ← propList \∪ \{(xp, op, s)\}
12: endforeach
13: endforeach
14: endforeach
15: return mappedObjProp \∪ propList
16: end procedure

Linguistic Mapping: The proposed linguistic mapping technique exploits the model presented in [22] built using the Word2Vec (W2V) algorithm [23] pre-trained on Google News dataset (about 100 billion words) [24]. In the rest of the work, we refer to this W2V pre-trained model whenever we use the expression W2V model. The linguistic mapping technique consists of the following five steps.

a) Initialization: This step loads the W2V model and outputs its unique terms as a vocab list and its similarity vector (i.e., the keyed vector representation of the terms).

b) Pre-processing the specifications: We assume that the XSD specification represents the knowledge as a set of ComplexTypes containing Elements and Attributes, along with their types, which can be either DataType or ComplexElement. On the other hand, the ontology represents knowledge as a set of Classes and related Properties. Properties can be either DataTypeProperty or ObjectProperty. A Property corresponds to a relation between its domain (represented by a Class) and its range (which can be a Class or a DataType). Inspired by the mapping rules introduced in [11], we designed a set of transformation rules presented in Table I. Then, this step parses the XSD and the ontology files and builds three sets of transformation rules presented in Table I. Lines 4-11 of Alg. 1 show the steps to obtain these sets. The sets extracted from the XSD file are xCl, xObPr, xDtPr, where xCl (resp., xObPr, xDtPr) is the set of candidate terms to be mapped to Classes (resp., ObjectProperties, DataTypesProperties) in the ontology. Concerning the ontology, instead, oCl (resp., oObPr, oDtPr) is the set of terms corresponding to Classes (resp., ObjectProperties, DataTypesProperties) in the ontology.

In the following steps (which are encapsulated in the application of the W2VlinguisticMap function on lines 14-16), mappings between pairs of terms from the various sets are created, depending on their nature, using a linguistic approach.

c) Finding n similar terms: The W2V model is applied separately to the six sets of terms to get n similar words for each term, where n is a configuration parameter (a positive integer). We tested various values for n (3, 5, 10, 20) to find a good balance between accuracy and efficiency of the approach, and we finally settled on n = 3. For instance, if x and y are the number of terms from the two specifications (XSD file and ontology), respectively, after obtaining n similar terms using the W2V model, the resulting matrices will have size x · (n + 1) (for the XSD file) and y · (n + 1) (for the ontology)—notice
that each original term is also included. Table II provides an example of the resulting matrix. The left-most column shows the terms from each specification, whereas the others show the words suggested through the W2V model for each term.

$$\text{(A): W2V suggestions for terms in the first specification}$$

<table>
<thead>
<tr>
<th>Term</th>
<th>SimilarTerm1</th>
<th>SimilarTerm2</th>
<th>SimilarTerm3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Itinerary</td>
<td>Itinerary</td>
<td>Itineraries</td>
<td>CruiseTour</td>
</tr>
<tr>
<td>EffectiveDeparture</td>
<td>Departs</td>
<td>SuccessionPlan</td>
<td>DepartsArrives</td>
</tr>
</tbody>
</table>

$$\text{(B): W2V suggestions for terms in the second specification}$$

<table>
<thead>
<tr>
<th>Term</th>
<th>SimilarTerm1</th>
<th>SimilarTerm2</th>
<th>SimilarTerm3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trip</td>
<td>Trip</td>
<td>Travels</td>
<td>FliesInto</td>
</tr>
<tr>
<td>departureTime</td>
<td>departing</td>
<td>abruptdeparture</td>
<td>departures</td>
</tr>
</tbody>
</table>

d) Match terms in the first specification to those in the second specification: In this step, each term from each matrix produced in step (c) for the first specification is matched to the terms in the corresponding matrix for the second specification. Therefore, the matrix obtained for set $xCI$ is matched to the one of $oCI$ (within the W2VlinguisticMap invocation on line 14), and similarly for the other sets of terms. The W2V similarity vector (discussed in step (a)) is used to compute the cosine similarity [25] ($CS$) for each pair of terms $\langle \text{MatTermS1}, \text{MatTermS2} \rangle$, where $\text{MatTermS1}$ (resp., $\text{MatTermS2}$) is a term of the matrix obtained for the first (resp., second) specification (notice that the matrices include the original terms retrieved from the specifications). $CS$ ranges from 0 to 1, where the higher the value, the higher the similarity. The resulting triplets have the form $\langle \text{MatTermS1}, \text{MatTermS2}, CS \rangle$. We set 0.5 as cut-off threshold, and consider a triplet as a potential generator of a mapping (according to the rule described in step (e)) if its $CS$ value is greater than or equal to the threshold.

e) Count number of matches: In this step we take each triplet $\langle \text{MatTermS1}, \text{MatTermS2}, CS \rangle$ obtained in step (d), and we trace it back to its pair of original terms $\langle \text{TermS1}, \text{TermS2} \rangle$ from the first and second specifications, respectively. For example, consider again Table II. Imagine that, after step (d), we have a triple $\langle \text{Itinerary, Trip}, 0.677 \rangle$. We trace it back to the original pair of terms $\langle \text{Itinerary, Trip} \rangle$, and we increase by 1 the counter of the number of matches between Itinerary and Trip. In this way, for each pair of original terms $\langle \text{TermS1}, \text{TermS2} \rangle$ we count the number of matches, and we compute the similarity value $CS_{ts1,ts2}$ for the pair as the average of the $CS$ values of the triples $\langle \text{MatTermS1}, \text{MatTermS2}, CS \rangle$ that trace back to it. At the end of this step, we produce a set of triples $\langle \text{TermS1}, \text{TermS2}, CS_{ts1,ts2} \rangle$.

Step (e) concludes the linguistic part of the mapping procedure. The rest of the algorithm (lines 19-27), which is explained next, refines and extends the suggestions by exploiting the structure of the two specifications.

**Structural Mapping:** The proposed structural mapping technique relies on the rules presented in Table I. More precisely, we use the structure of the ontology as guidance to further refine the mappings returned by the linguistic mapping. First of all, notice that the results produced by the linguistic mapping are stored in three sets of triples of the form $\langle \text{TermS1}, \text{TermS2}, CS_{ts1,ts2} \rangle$ named mappedClass, mappedObjProp and mappedDataProp, where $\text{TermS1}$ and $\text{TermS2}$ are names of Classes, ObjectProperties, or DataProperties, depending on the set. Notice that, for the sake of structural mapping, we consider as Classes, ObjectProperties and DataProperties also elements from XSD specifications when they match the rules of Table I (e.g., a ComplexType is considered, for structural mapping purposes, as a Class).

In this step we consider that each specification defines triples of the form $\langle \text{Domain, ObjectProperty, Range} \rangle$. In the following, we indicate a triple from the first (resp., second) specification as $\langle \text{DomainS1, ObjectPropertyS1, RangeS1} \rangle$ (resp., $\langle \text{DomainS2, ObjectPropertyS2, RangeS2} \rangle$).

We perform the following refinements of the mappings.

(i) Suggest properties if domains and ranges match: If, in mappedClass, DomainS1 is mapped to DomainS2 and RangeS1 is mapped to RangeS2, then triple $\langle \text{ObjectPropertyS1, ObjectPropertyS2, CS_{ops1,ops2}} \rangle$ is added to set mappedObjProp, where $CS_{ops1,ops2}$ is the average of the confidence scores of the mappings between domains and ranges—i.e., pair $\langle \text{ObjectPropertyS1, ObjectPropertyS2} \rangle$ is suggested with confidence score $CS_{ops1,ops2}$. This step is performed by the procedure invoked at line 19 of Alg. 1; in particular, the addition of each single new pair is performed by lines 9-13 of Alg. 2.

(ii) Suggest domains (resp., ranges) if properties and ranges (resp., domains) match: If ObjectPropertyS1 is mapped to ObjectPropertyS2 in mappedObjProp and RangeS1 is mapped to RangeS2 in mappedClass, then pair $\langle \text{DomainS1, DomainS2} \rangle$ is suggested with confidence score $CS_{ds1,ds2}$, where $CS_{ds1,ds2}$ is the average of the confidence
Algorithm 3 New Classes Based on Properties and Classes

1: procedure ADDCLASSESFROMCLASSESANDOBJPROP
2: input: mappedObjProp, mappedClass, xCl, oCl
3: output P: set of triples (xt, ot, s), xt ∈ X, ot ∈ O, s:Confidence score
4: rangeList, domainList ← ∅
5: foreach (xc1, oc1, s1) ∈ mappedClass do
6:    foreach (xpj, opj, s2) ∈ mappedObjProp do
7:      if xc1 = xpj, ComplexType & xp = xpj, Type &
8:         oc1 = opj, Domain & oc = opj, Range then
9:         s ← (s1 + s2)/2
10:        rangeList ← rangeList ∪ \{(xc, oc, s)\}
11:     end if
12: end foreach
13: end foreach
14: return mappedClass ∪ rangeList ∪ domainList
15: end procedure

Algorithm 4 New Classes Based on Properties

1: procedure ADDCLASSESFROMOBJPROP
2: input: mappedObjProp, mappedClass, xCl, oCl
3: output P: set of triples (xt, ot, s), xt ∈ X, ot ∈ O, s:Confidence score
4: rangeList, domainList ← ∅
5: foreach (zp, op, s) ∈ mappedObjProp do
6:    foreach (xci, oc) ∈ xCl × oCl do
7:       if xci = zp, ComplexType & oc = op, Domain &
8:          xci = zp, Type & oc = op, Range then
9:          s ← (s + 0.6)
10:         domainList ← domainList ∪ \{(xci, oc, s)\}
11:        rangeList ← rangeList ∪ \{(xci, oc, s)\}
12:     end if
13: end foreach
14: end foreach
15: return mappedClass ∪ rangeList ∪ domainList
16: end procedure

(iii) Suggest domains and ranges if properties match: In this case, if ObjectPropertyS1 is mapped to ObjectPropertyS2 in mappedObjProp, we suggest pairs ⟨DomainS1, DomainS2⟩ and ⟨RangeS1, RangeS2⟩, both with confidence score that is 60% of that of the mapping between the properties (i.e., that is equal to 0.6 · CS_{opS1,opS2}). This step is performed by the procedure invoked at line 25 of Alg. 1 (see also Alg. 4).

Although the algorithms presented above assume that one specification is given as an XSD file, and the other as an ontology, indeed they have been adapted to also work when the input specifications have the same format (i.e., they are both XSD files, or both ontologies). More precisely, if both inputs are XSD files, then the algorithm simply performs the same pre-processing step explained in point (b) of the linguistic mapping on both files, to extract a set of "candidate classes" xCl₁, xCl₂ from each file, which are then used in the rest of the algorithm instead of xCl and oCl (similarly for object and data properties).

B. Annotation Generation

The suggested mappings, following a review of the user, are sent to the the annotation generation module for the final step of the process (see Fig. 2). The annotation generation module is composed of two pipelines, one for each type of annotations supported. More precisely, it can produce either Java annotations compatible with the approach presented in [7], or YARRRML rules compatible with the converter presented in [6]. Since the annotation generation step is tightly linked to the conversion approach depicted in Fig. 1, it assumes that the first specification is given in terms of an XSD file, whereas the second is an ontology. The rest of this section provides a brief description of the two pipelines.

Java annotation pipeline: In this case, the module analyzes each suggested mapping, and produces a corresponding Java annotation. More precisely, it first determines whether the mapping concerns Classes of Properties. Depending on the case, it fills the appropriate template (i.e., @Rdfs\(<\text{TYPE}\>\("<\text{ONTOLOGY NAME}\>\:<\text{TARGET TERM}\>\)\)) and outputs a suitable annotation. For example, lines 2 and 4 in Fig. 3 show a pair of annotations for Class and Property mappings, respectively. For instance, if the term “GeoPoint” (in the XSD specification) is mapped to term “GeoCoordinates”, which is a Class in the reference ontology (in this case, the IT2Rail Ontology [26], see also Sect. V), the annotation will be @RdfsClass("IT2Rail: GeoCoordinates"). Then, the annotation generation module uses the JAXB package [27] to create, from the XSD file, the Java classes to be annotated. In the final step of the pipeline, the module, for each mapped term of the XSD specification, parses the Java files to find the term’s declaration, then it inserts the corresponding annotation appropriately (see lines in bold in Fig. 3). Finally, the user receives a zipped folder containing the annotated Java classes.

YARRRML generation pipeline: YARRRML [28] is a human-readable text-based representation for declarative Linked Data generation rules. The YARRRML generation module first identifies the structural relationships between the terms in the XSD specification. More precisely, the module extracts, for each term of the XSD file corresponding to a Class, its Properties and stores those for which there is a suggestion in the confirmed mappings. Next, according to YARRRML’s syntax, the module generates the appropriate prefixes and mappings blocks. The prefixes block contains the required namespaces (e.g., the ontology’s namespace). In the mappings block, instead, the module defines, for each term corresponding to a Class in the XSD specification, the following three elements: (i) data source location, (ii) subjects’ generation, and (iii) predicate-object annotations. As the result, the user receives the YARRRML declarations in a .yml file.

433
@RdfsClass("IT2Rail:GeoCoordinates")
public class GeoPoint extends FSMID
{
@RdfProperty(propertyName = "IT2Rail:hasLatitude")
@XmlAttribute(name = "Latitude", required = true)
protected BigDecimal latitude;

Fig. 3. Example of Java annotations for Class and a Property (in bold).

IV. TOOL

The current version of the SMART prototype is comprised of two containerized components: a RESTful API and a web server hosting a front-end built with Angular [29], both communicating using JSON standard.

The API is designed to allow multiple simultaneous user requests by relying on the FastAPI [30] framework; the asynchronous environment coupled with the NginX [31] web server allows the tool to manage high throughput of parallel requests. Although NginX can handle up to thousands of requests at once, the API is limited by the number of resources the mapping process requires (e.g., in our experiments, each run was using up to 8GB of RAM): requests that would require SMART to exceed the available processing power are enqueued for when the tool will become available again.

After uploading and selecting the specification files to be mapped, the user can send a mapping request to the server. Every incoming request is given a unique identifier and is handled separately by the tool, allowing it to store and retrieve results asynchronously. Once a request has been parsed, the tool spawns a separate process to handle the computationally heavy mapping process and relinquishes the control until this operation is completed. The process itself can take several minutes to complete.

Once the mapping phase is completed, the user receives a selection of up to 3 most-similar ontology terms for each source term in the XSD file, each associated with its CS value (see Sect. III-A) discretized as high ($CS \geq 0.75$), medium ($0.30 \leq CS < 0.75$) and low ($CS < 0.3$) confidence. The user can either confirm the suggested mapping, select one of the alternatives, or add a choice of their own making as they see fit, as shown in Fig. 4. In this case, SMART has selected the ontology term Trip as a possible mapping for the XSD term Itinerary with a medium CS rating. The “Other” input field is used to create a new alternative if the user is not satisfied with SMART recommendations. In addition, Fig. 4 shows that, for the GeoPoint term of the XSD file, SMART suggested a term (PointOfInterest), but the user decided to change the mapping to GeoCoordinates. Alternatively, the user can choose the automatic mapping process, in which the tool automatically selects the term with the highest CS value for each pair. After the selection has been made in either way, the user can proceed with the annotation generation phase based on the pipeline selected at the beginning of the mapping process, then the annotated files can be downloaded.

V. VALIDATION

We evaluated the effectiveness of the approach on a few case studies involving specifications from the transportation domain. In particular, this section focuses on the accuracy of the Mapping Algorithm (see Sect. III-A), which is the core of the approach, in terms of its ability to suggest meaningful mappings. More precisely, we took five specifications (XSD files from NeTEx [32], FSM [33] and Neptune [34], and the IT2Rail [26] and Transmodel [35] ontologies) from the transportation domain, we used the Mapping Algorithm to generate suggested mappings between various pairs of specifications, then we manually evaluated the accuracy of the suggestions output by the algorithm (before the review of the user).

Table III collects the basic information (type of specification—XSD file or ontology—and number of terms) about the five specifications used, which are briefly introduced in the following. The IT2Rail ontology was created in the project by the same name [26] and it is at the basis of the ontology that is currently being developed within the Shift2Rail Innovation Programme 4. Transmodel (short for “Public Transport Reference Data Model”) is a European Standard that covers various areas of the transportation

<table>
<thead>
<tr>
<th>Specification</th>
<th>Type</th>
<th>Number of terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeTEx XSD</td>
<td>XSD</td>
<td>91</td>
</tr>
<tr>
<td>FSM XSD</td>
<td>XSD</td>
<td>113</td>
</tr>
<tr>
<td>IT2Rail Ontology</td>
<td>Ontology</td>
<td>543</td>
</tr>
<tr>
<td>Transmodel Ontology</td>
<td>Ontology</td>
<td>231</td>
</tr>
<tr>
<td>Neptune XSD</td>
<td>XSD</td>
<td>89</td>
</tr>
</tbody>
</table>

Fig. 4. SMART term selection example.

Available from github.com/alexander-nemirovskiy/s2r_mapping_ui.git.

A short video of the tool is available at github.com/alexander-nemirovskiy/s2r_mapping_ui/raw/master/docs/SMART_demo.mp4.
domain, such as network topology representation, scheduling, operation monitoring, fare management, etc. Recently, a full-fledged ontology [35], which we have used for our evaluation, has been defined starting from the Transmodel standard. NeTEx [32] is a CEN technical standard for exchanging public transport schedules and data. NeTEx is a large standard, which is divided into three parts, each covering a subset of Transmodel standard. For our experiments, we considered a subset of NeTEx, focusing on the description of the infrastructure (stop points, vehicles, etc.). FSM (short for “Full Service Model”, [33]) is a standard for representing information about ticketing and reservations in a heterogeneous transport environment. Finally, Neptune [34] is the reference format used in France to exchange information concerning public transport (itineraries, timetables, etc.).

Each test case examines a different combination of specifications. Table IV lists the combinations that we tried. For each test case, we carefully assessed the output results to determine the accuracy of the mapping suggestions. For this purpose, we relied on the documentation of each specification describing the terms in the dataset. We categorized each suggested pair as Correct, Incorrect, Ambiguous, or Unfeasible.

![Figure 5](image-url)

Fig. 5. Detailed results of the evaluation, where the numbers in the bars correspond to the number of pairs in each category, for each test case.

The first two categories are self-explanatory. A pair is deemed Ambiguous if there is not enough information about the meaning and usage of a term in the specification to evaluate the correctness of the suggestion. A mapping is considered Unfeasible if no equivalent representation of the term in the first specification is available in the second one. Given this categorization, the accuracy of the results of each test case was computed as the percentage of the Correct mappings over the sum of Correct and Incorrect ones. Both the Ambiguous and the Unfeasible categories are excluded from the computation as they do not provide a meaningful contribution for it, since the pair either lacks a clear definition, or there is no alternative for the first term in the second specification.

Table IV presents, for each test case, the corresponding accuracy, and the time that it took to generate the suggested mappings using the procedure of Sect. III-A. For our experiments, we deployed the tool on a general-purpose Amazon EC2 instance with 32GB Memory, 8 vCPU 3.0 GHz Intel Xeon processor, and up to 1Gb/s connection speed. The duration of the mapping generation process depends on the size of the input specifications; however, as shown in Table IV, no experiment took more than 6 minutes. For completeness’ sake, Figure 5 provides, in addition to a graphical representation of the share of each category of mapping for each test case, the number of elements in each category. On average, SMART’s accuracy is 66.4%, ranging from 48% to 78% (recall that, when computing the accuracy, we do not count Ambiguous and Unfeasible mappings in the denominator of the ratio).

VI. CONCLUSION

This paper presented a tool-supported approach to suggest mappings between terms of separate data specifications; this is the basis for converting data between different data formats, which is a core enabler of interoperability in heterogeneous Systems of Systems. The approach, which is also able to automatically create, from selected mappings, annotations compatible with the conversion mechanisms introduced in [6], [7], has been validated on various test cases from the transportation domain, showing promising results.

In the future, we will refine both the underlying suggestion mechanism and the supporting SMART tool. In particular, we plan to explore the possibility of using domain-specific models (e.g., tailored to the transportation domain) for the linguistic mapping part of the procedure, instead of the general-purpose one used in this work; this would improve the accuracy of the first step of the procedure. In addition, we plan to extend the structural mapping part of the algorithm with richer rules, able to handle a wider range of features of XSD files.

Acknowledgment

This work was supported by Shift2Rail and the EU Horizon 2020 research and innovation programme under grant agreement No: 826172 (SPRINT).

References

Incorporating Presuppositions of Competency Questions into Test-Driven Development of Ontologies

Jedrzej Potoniec\textsuperscript{1,2}, Dawid Wisniewski\textsuperscript{1} Agnieszka Ławrynowicz\textsuperscript{1,2}

\textsuperscript{1} Faculty of Computing and Telecommunications, Poznan University of Technology, Poland
\textsuperscript{2} CAMIL Center for Artificial Intelligence and Machine Learning, Poznan University of Technology, Poland

E-mail: \{jpotoniec, dwisniewski, alawrynowicz\}@cs.put.poznan.pl

Abstract

Ontology authoring is a complicated and error-prone process since the knowledge being modelled is expressed using logic-based formalisms, in which logical consequences of the knowledge have to be foreseen. Many approaches intended to make this task easier, use competency questions (CQs), being questions expressed in natural language to trace both the correctness and completeness of the ontology at a given time. However, CQs hold so-called presuppositions that have to be satisfied by the ontology to obtain meaningful answers from CQs. Moreover, CQs have to be expressed using a formal language, like ontology query language (SPARQL-OWL), to query the ontology. In this paper, we propose an extension of test-driven ontology development approach by formalization of presupposition satisfaction tests in terms of SPARQL-OWL queries, as well as providing translations of CQs into SPARQL-OWL queries if presupposition tests are passed. We provide a detailed description of the proposed framework and how to incorporate such tests in the workflow of test-driven development of ontologies. It is the first framework available for formalization of SPARQL-OWL queries out of CQs with their presupposition tests.

1. Introduction

Competency Questions (CQs) are questions expressed in natural language, which aim to define the scope of an ontology as part of the ontology requirements specification [14]. They are, subsequently, formalized using a relevant language, such as SPARQL [6], to automatically validate whether the ontology meets the requirements [7, 4, 2].

Ontologies can be expressed in vastly varying modeling styles, using varying modeling patterns [9], and varying features of an ontology modeling language such as the Web Ontology Language (OWL) [11]. Consider the following CQ: What software has an open source licence? If one models different applications as different individuals, one can easily construct a suitable query in SPARQL [6] to list all instances of the class Software. However, if one models different applications as different classes, one must query for subclasses of the class Software instead.

The particular formalizations to express the tests have also varied, depending on factors such as features of the modeled domain, and included axioms and axiom sets [2], SPARQL-OWL [7, 16] or even methods based on instance assertions mimicking mock-objects [7].

In this paper, we consider formalising CQs using SPARQL-OWL [8], a variant of SPARQL with an OWL 2 DL entailment regime, that enables to handle the formal semantics of ontologies expressed in OWL 2 and assumptions underlying reasoning in OWL. Specifically, OWL makes an assumption, different from the Closed World Assumption made in databases: the Open World Assumption states that if a fact cannot be proved, it does not necessarily mean that it is false. Firstly, SPARQL-OWL will allow us to use features of the OWL and OWL reasoning. Secondly, it will also allow us to express queries with answers different than true/false, e.g., SELECT queries expressing list questions.

CQs not only encode explicit intents of their creators, but also implicit assumptions, so-called presuppositions [13, 4]. The notion of presuppositions comes from linguistic pragmatics, where a presupposition of a statement is a proposition whose truth is a precondition to assess whether the sentence is true or false. If the presupposition does not hold, then the sentence cannot be assessed either as true or false. Considering a question sentence, a presupposition has to be true in order for the question to have an answer. For example, the question What software has an open source licence? presupposes that software can have a licence, and if not, then the question cannot be answered.

In this paper, we provide a methodology for formalizing CQs into their corresponding formalized queries and associated presupposition queries as testing artefacts with the interpretation of their results. We consider the following sce-
nario: 1) Domain expert states CQs 2) Ontology engineer(s) create(s) the ontology 3) During ontology development, CQs are translated into SPARQL-OWL queries, so queries and answers can be obtained and verified after vocabulary is modelled. The scenario is analogous to software engineering, where one states unit tests before the software exists, and then, during development, these tests measure the quality of the software and may help to decide when the authoring process is complete.

We aim to address the following research questions: RQ1: What does it mean that a CQ is answerable? RQ2: What does it mean for CQ-driven ontology authoring that a presupposition is satisfied or not when it comes to testing? RQ3: How to handle presuppositions in the workflow of test-driven development of ontologies?

Our contributions are as follows: (a) a formalization of presuppositions using SPARQL-OWL ASK queries, (b) a model for testing list questions that considers presupposition tests with their interpretation, (c) incorporating presupposition tests into the workflow of test-driven ontology engineering, (d) a dataset of SPARQL-OWL queries enhanced with their presupposition queries.

The remainder of the paper is structured as follows. Sect. 2 describes related work. Sect. 3 introduces the formalization of presuppositions as SPARQL-OWL ASK queries, a model of testing, and describes the incorporation of presuppositions into the test-driven ontology engineering workflow. We conclude in Sect. 4.

2. Related Work

2.1. Analysis of CQs

Ren et al. [13] analysed CQs and determined patterns in the form of CQ archetypes (e.g., “Which [CE1] [OPE] [CE2]?”) containing placeholders for presupposed ontology elements. Bezerra et al. [1] also proposed CQ patterns with placeholders for ontology elements, e.g. “Does <class>+<property><class>?”, functioning as Controlled Natural Language.

Wiśniewski et al. [16] and Potoniec et al. [12] analysed the natural language text of CQs itself, and a subsequent step of semantic analysis in order to find patterns. Next, they analysed the relation between the found CQ patterns and their respective SPARQL-OWL signatures (abstract representations of SPARQL-OWL query meaning), which revealed that one CQ pattern may be realized by more signatures and vice versa. Wiśniewski et al. [17] proposed a machine learning based approach to parse CQs with a model trained on over 46,000 automatically generated CQs.

Fernández-Izquierdo et al. [5] collected a corpus of ontological requirements annotated with lexico-syntactic patterns (named CORAL). The lexico-syntactic patterns have OWL constructs associated to them. These constructs were extracted from the ontology design patterns (ODPs) associated with the given lexico-syntactic pattern.

2.2. Test-driven development of ontologies

There are several tools proposed for test-driven development (TDD) for ontologies. Tawny-OWL [15], an ontology development framework, provides predicate functions to query the reasoner, and its answer is true/false. TDDOnto is a Protégé plugin which avails of the Protégé’s syntax and uses the reasoner through the OWL API [7]. TDDOnto2, which extends TDDOnto, rigorously proves the correctness of the testing algorithms of TDDOnto, Tawny-OWL, or SConE [3]. It generalises the algorithms of Keet and Lawrynowicz [7] to cover any OWL 2 class expression in the axiom under test.

When it comes to testing results, all the mentioned tools except TDDOnto2 give only limited information about the result of any test, being pass/fail in Tawny-OWL and SConE. More precisely, only “axiom entailed” by the ontology is a pass and all the others statuses are test failures. TDDOnto also reports missing vocabulary. TDDOnto2 specifies failure statuses more precisely, being either “inconsistent”, “incoherent”, or “absent”.

All TDDOnto2 tests are expressed using axioms which can be tested in terms of their truth values availing of a reasoner, with a purpose of checking whether the knowledge encoded directly via the axiom is already covered in the ontology (is entailed). However, when it comes to CQs, which are questions associated often with ‘gold standard’ answers, they are more naturally expressed as queries, such as list queries (i.e., queries with a result being a list of objects). Therefore, in this paper, we explore such direction.

Another aspect of the mentioned tools is that they do not consider presuppositions. Indeed, binary questions (such as on the truth value of an axiom) do not have presuppositions [13] since they simply ask whether there is an answer satisfying the constraint. In this paper, contrary to the mentioned works, we consider presuppositions as we deal with SPARQL-OWL SELECT queries.

3. Incorporating presuppositions into TDD for ontologies

3.1. Presuppositions in CQs

Linguistic research on pragmatics reveals that a list question, starting with WH-words like what, which etc. always makes a presupposition that some object(s) fulfil the predicate of the question [10]. A presupposition can be generated by replacing the WH-word with the corresponding indefinite pronoun, e.g., Who left the door open? presupposes Someone left the door open [10]. It is possible to deny
the presupposition, e.g., No-one left the door open [10].

From this we infer that a CQ always assumes, either explicitly or implicitly, some domain for its answer. Moreover some elements of the domain must be capable of fulfilling the predicate (a positive presupposition), yet the elements of the domain not necessarily fulfil the predicate (a negative presupposition).

3.2. Model of testing with presuppositions

Let us denote by $O$ an OWL 2 ontology [11], and by $C, D$ a named class or a class expression. Denote by $Q$ a formalization of a CQ in the form of a SPARQL-OWL query, and a positive presupposition query by $PQ^+$, and a negative presupposition query by $PQ^-$. Below, we formalize presupposition tests availing of SPARQL-OWL queries plus the interpretation of their results.

A presupposition query $PQ$ is a SPARQL-OWL ASK query with only the following basic graph pattern (BGP) in the WHERE clause: $C \text{ rdfs:subClassOf } \text{owl:Nothing}$. Note, that we use such formulation since there is no direct syntax for satisfiability checking.

Definition 3.1 (Presupposition test). Let $\Psi(PQ)$ denote the solution sequence, as defined by [6], of the presupposition query $PQ$ under the OWL 2 DL entailment regime over the ontology $O$. If $O \models C \text{ rdfs:subClassOf } \text{owl:Nothing}$, then $\Psi(PQ) \neq \emptyset$ and the answer to $PQ$ is true, denoted $\mu(PQ) = \text{true}$, meaning the presupposition is not satisfied. Otherwise, $\Psi(PQ) = \emptyset$ and the answer to $PQ$ is false, denoted $\mu(PQ) = \text{false}$, meaning the presupposition is satisfied.

Furthermore, we define the model of testing for SPARQL-OWL SELECT queries $Q$ as for those queries that have presuppositions. SPARQL-OWL ASK queries $Q$ as corresponding to binary questions do not have presuppositions [13] since they simply ask whether there is an answer satisfying the constraint.

Let us now introduce the model of testing. We start from a CQ, for instance Which pizzas contain chocolate?. This induces a positive (informal) presupposition There may exist pizzas with chocolate, and a negative presupposition There may exist pizzas without chocolate. Both the CQ, and associated presuppositions are formalized as SPARQL-OWL queries.

Any list question (formalized as SPARQL-OWL query) can be considered as restricting a certain class expression $C$ with another class expression $D$, i.e., a question about $C$ and $D$. Then positive presupposition means that there are some objects that are both $C$ and $D$, and if negative presupposition is satisfied, it means that there are objects that are $C$ but not $D$. Non-fulfillment of a positive presupposition means that the ontology determines the answer: the intersection of $C$ and $D$ is necessarily an empty set; failure to meet negative presupposition means that the ontology determines the answer: the intersection of $C$ and $D$ is equivalent to $C$.

If answers to all presuppositions of $Q$ are false, then we can ask the query $Q$ and interpret the obtained result.

Definition 3.2 (Model for testing (SELECT query)). Given a consistent and coherent ontology $O$, a SPARQL-OWL query $Q$ asking about the intersection of the class expressions $C_i$ and $D_i$, and its positive presuppositions $PQ^+_i$ (there are objects being both $C_i$ and $D_i$), a corresponding negative presuppositions $PQ^-_i$ (there are objects being $C_i$, but not $D_i$), ($i=1...n$), then the result of testing $Q, PQ^+_i$, and $PQ^-_i$ against $O$ is:

$$
\text{test}_O(Q) = \begin{cases} 
\mu(Q) = C & \text{if } \exists i \mu(PQ^+_i) = \text{true (i.e., unsat.)} \\
\mu(Q) = \emptyset & \text{if } \exists i \mu(PQ^-_i) = \text{true (i.e., unsat.)}
\end{cases}
$$

compute the answer to $Q$ if $\forall i \mu(PQ_i) = \text{false}$

3.3. Presuppositions in TDD workflow

Test-driven approach to ontology authoring has been shown to be theoretically and technologically a worthy solu-
tion and the recommended TDD ontology authoring workflow has been proposed [2]. Here, we extend the workflow with new additions to incorporate checking question answerability, presuppositions and list questions. The extended workflow is depicted in Fig. 1. Grey boxes represent the steps of the extended TDD workflow, which we incorporate and focus on in this paper. In particular, to determine query answerability, we not only need to check whether there is an answer to a query consistent with our intention, but whether the query can be constructed at all, including checking whether there is relevant vocabulary. Then, positive and negative presupposition tests serve to further check whether constructing a list query is meaningful.

Only after these steps, one can construct a SPARQL-OWL query out of a natural language CQ.

3.4. Dataset

In https://tinyurl.com/3v4rfp6f we provide a dataset consisting of SPARQL-OWL CQ query templates and their corresponding SPARQL-OWL query templates for presuppositions. The dataset is an extension of a preexisting dataset of SPARQL-OWL formalization of CQs [12, 16]. In the templates any IRIs referring to any concrete ontology are replaced by placeholders denoted by angle brackets and the same placeholder in the CQ query template and in the presuppositions templates should be replaced by the same value during materialization.

4. Discussion & conclusions

Answering RQ1, the notion of answerability of a CQ has two levels. It may seem that it is sufficient for the necessary vocabulary to be present in the ontology. While this always yields some query, and thus some answers, it fails to consider the reason for the answers, which may go against the intent of the CQ. One must thus consider the presuppositions inherent to the CQ and test them beforehand. Only when the presuppositions are satisfied the answer to the query follows the intent of the CQ. We thus claim that a CQ is (meaningfully) answerable if the necessary vocabulary is present and all the presuppositions are satisfied.

Addressing RQ2, we introduced in Sect. 3.2 the notion of a presupposition query. We formalized the way of handling such a query in order to create a presupposition test and offered guidelines to extract presuppositions from a CQ formalized as a SPARQL-OWL query.

Regarding RQ3, we extended the workflow of TDD for ontologies to incorporate presuppositions. In Sect. 3.3 we explained that an unsatisfied presupposition denotes that an answer for the CQ is predefined in a way that is incompatible with the intent of the query.

Being able to automatically provide correct SPARQL-OWL query recommendations and their presupposition tests as formalizations of ontology competency questions for a given ontology is a promising idea which can lead to reduction of time required to author the ontology. Using an already existing dataset of CQs and their translations to SPARQL-OWL helped us to get the first insight into the problem. We hope that our model for extending TDD for ontologies with presupposition tests, together with a new dataset, will incent further research into more commonsense-aware knowledge engineering.

References

Ride Hailing Service Demand Forecast by Integrating Convolutional and Recurrent Neural Networks

Zinat Ara  
Department of Information Sciences and Technology  
George Mason University  
Fairfax, VA, USA  
zara@gmu.edu

Mahdi Hashemi  
Department of Information Sciences and Technology  
George Mason University  
Fairfax, VA, USA  
mhashem2@gmu.edu, ORCID: 0000-0003-0212-0228

Abstract—Ride hailing services, such as Uber, Lyft, and Grab have become a major transportation mode in the last decade. Current ride demand is one of the major factors in such services’ pricing algorithm. Therefore, forecasting future travel demand for such services is essential to both drivers and riders. This study constructs a deep learning based model for ride hailing demand forecast aiming to achieve high accuracies in solving similar problems. This study attempts to address a limitation in existing ride hailing demand prediction models, where the area is divided into a rectangular grid and all travel demand forecasts are made between rectangular cells, rather than city neighborhood zones. The proposed model forecasts travel demand between city neighborhood zones. The forecast model integrates convolutional and recurrent neural networks and forecasts the demand for each pickup-destination pair for a particular hour, during the next day, by observing the demand over the past two weeks for that particular hour. Our experiments with a real-world hire vehicle dataset in New York City showed that the proposed model outperforms the CNN and LSTM models up to 18.41 % in RMSE and 22.65 % in R² values.

Keywords: travel demand prediction; deep learning; convolutional neural network; recurrent neural network

I. INTRODUCTION

Companies, such as Uber, Lyft, Juno, Gett, Grab that allow passengers to request rides from mobile applications are known as ride-hailing service companies. Online ride-hailing services have rapidly grown in popularity in recent years. Receiving a correct approximation of travel demand at a certain time can help a passenger schedule their future trips more efficiently. On the other hand, drivers employed by the ride hailing services can choose to work at times with high demand to maximize their earnings. This paper proposes a deep learning method combining convolutional neural network and bidirectional long short term memory (biLSTM) for predicting travel demand of ride-hailing services.

Space and time are the two indispensable dimensions of urban ride-hailing demand prediction. For instance, urban ride-hailing is always strongly linked between the public transport areas and the tourist regions, and between the residential and the industrial regions. In the peak hours of workdays morning and the peak hours of workdays evening, requests from residential and industrial regions increases significantly. In addition to the association between space and time, changes of urban ride-hailing demand are also affected by many other factors including weather, holidays, and unforeseen incidents such as sport events or national occasions. There have been extensive studies in traffic data prediction, including traffic volume, taxi pick-ups, and traffic in/out flow volume. With the fast developing of machine learning models, prediction models based on neural network is getting more attention. Existing methods on traffic prediction have only considered spatial relation (e.g., using CNN) or temporal relation (e.g., using LSTM) independently and state-of-art results are achieved in the work compared with the traditional statistical learning methods. The main advantage of CNNs that make them suited to forecasting time series is the ability to use filters to compute dilations between each cell. Considering the spatial relation which is the size of the space between each cell, allows the neural network to better understand the relationships between the different observations in the time series. In this paper, we are combining bidirectional LSTM with CNN for predicting ride-hailing demand. Our work is different from the existing ones [1, 2, 3] as we are not using longitude-latitude to create grids for defining the location and the method does not require any graphical representation or image data as input. A rectangular division of a map can result in grids which fall into two different regions or have a majority part of it outside the land area. In either of these cases the demand calculation can suffer a level of inaccuracy. The study divides the total space into regions using the NYC Taxi Zones dataset which creates partitions based on NYC department of city planning’s neighborhood tabulation areas. Pickup-destination pairs are created according to their regions and multiple one-dimensional arrays considering time period fit to be the input for our CNN. Applying filters, we convolve these one-dimensional matrices and then run biLSTM for each of them to predict ride-hailing demand. Most of the studies have applied unidirectional LSTM to handle similar problems but bidirectional will run the inputs in two ways and can understand the prediction context better [4].

II. RELATED WORK

Machine learning (ML) and prediction models are used by researchers to make transportation systems more intelligent [5, 6, 7, 8, 9, 10, 11, 12]. Lam and Liu [13] used the discrete choice model to analyze the correlation between dynamic pricing and waiting time in densely populated areas of New York. Gerte et al. [14] examined the demand for the ride-hailing service using
a panel based random effects model in order to capture both heteroscedasticity and autocorrelation effects. The major challenge of statistical approaches is a lack of predictive accuracy, particularly under a complex data environment with different data sources.

Travel demand data has periodicity, so historical travel demand is used to predict future travel demand. Ma et al. [15] utilized CNN on images of traffic speed for the speed prediction problem. In these methods convolution network is applied on the entire city without any partitions and predictions were done based on all the regions. As a result, taking account of unrelated regions for the prediction of the target area may downgrade the performance. LSTM itself is widely used to process time series data and traffic prediction. Yu et al. [16] applied long short-term memory network and autoencoder to capture the temporal dependency for predicting the traffic volume particularly for peak-hour and post-accident scenarios. However, the spatial relation is not measured in these scenarios.

To capture both spatial and temporal dependences simultaneously in one end-to-end training model, researchers have made numerous attempts in recent years. Shi et al. [17] proposed the conv-LSTM network, which combined CNN and LSTM in one sequence to sequence learning framework and the results showed that the conv-LSTM outperformed fully-connected LSTM on the basis of learning the complex spatio-temporal features. A study by Ke et al. [18] applied the random forest framework to select the exogenic variables, ranking their significance. In addition, the image intensity was examined which was retrieved from the map sequences of travel time rates using CNN and LSTM tools for short time passenger demand forecasting. Rodrigues et al. [20] presented a deep learning architecture merging the text information with time-series data and applied the approach to the problem of taxi demand forecasting in event areas.

Previous methods are mainly designed to predict the taxi demand in a specific region and ignore the importance of movement direction between different areas [21]. Rare works have been done on the prediction considering all combination including destination and inter-region demands. Liu et al. [22] aims at predicting the taxi demand between all region pairs in a future time interval. An approach contextualized spatial-temporal network is proposed and proved to be effective in predicting taxi demands both in origin and destination. However, region partition is done by grid based method and the spatial and temporal information of taxi demands has not been fully taken into consideration in this case. Instead of using traditional matrix Chu et al. [23] developed and manipulated origin-destination (OD) based tensor to represent OD flows and applied convLSTM model to predict demand. Grid wise division is applied for measuring OD tensor. Guo [24] proposed an integrated CNN-BiLSTM-Attention based model to predict taxi demand. Pengfeng et al. [21] divided the urban area into H x W grids based on the longitude and latitude. While calculating travel demand, they have considered both pickup and destination location to create a demand matrix. All of these studies divide area’s latitude and longitude into n x n geographical rectangles to consider the regions. But a region’s map is not rectangular shape and dividing it as this can result in grids falling into two different regions or have a majority part of it outside the land area. In these cases, demand calculation will be inaccurate.

This paper is considering pickup-destination pairs according to their regions and create multiple one-dimensional arrays for different time periods fit to be the input for our CNN. Then adding bidirectional LSTM layers for each of them to predict ride hailing demand which will run the inputs in two ways to better understand the prediction context.

III. DATA DESCRIPTION

One of our primary datasets is collected from NYC Open Data source which provides a wide range of traffic datasets in different formats. We have chosen 2018 (January-December) for hire vehicles trip data. These records are generated from the Ride Hailing Services (RHS) trip record submissions made by different commuter vehicles. The RHS trip records include fields capturing the pick-up date, time, and taxi zone location ID, which correspond with the NYC taxi zones open dataset. Each row represents a single trip in a hired vehicle service [25]. Secondly, NYC taxi zones data corresponds to the pickup and drop-off zones, or location IDs, included in the yellow, green, and RHS trip records (Uber, Lyft) published to Open Data. The taxi zones are based on NYC Department of City Planning’s Neighborhood Tabulation Areas (NTAs) and are meant to approximate neighborhoods, so one can visualize which neighborhood a passenger was picked up in, and which neighborhood they were dropped off in. This dataset provides the geolocations of neighborhoods where the output is multi polygons each representing different region with unique location ID.

![Figure 1 Pickup demand pattern in NYC in year 2018](image)

Figure 1 shows the number of requests along Y axis in 2018 for RHS in New York city from January to February based on pickup datetimes (X axis). From the pattern of the distribution with respect to time it is certain that passenger requests are higher in weekdays and comparatively lower in weekends. The features which have been considered for training in this study are timestamp, hour, demand, and weekend. These features were chosen for their higher impact over demand prediction and availability.
IV. METHODOLOGY

A. Non-grid partition

Most of the related studies have [21, 3, 22, 23, 24] used grid wise rectangular division method where they selected latitude and longitude value of a city and then divided into n×n geographical squares. For simplicity we define each square block unit as pixel. The overall travel demand about one city is reflected by each pixel’s demand. The grid wise division in geographical area is shown in Figure 2. But this method has few problems in following cases:

1. First problem will be the resolution; a pixel may contain regions with high demand and regions with low demand so they will combine in the same pixel and that pixel will basically show their average demand. The larger the pixel gets this problem becomes more severe (Figure 2).
2. A pixel may contain only regions with water, no land area. Demand will be zero in that pixel. These types of pixels cannot be disregarded in grid wise division method.
3. If a pixel includes both land and water area, that pixel will not provide the actual demand of the region. There is a high chance that demand counts might decreased for considering non-regional areas into the same pixel.

A geographical map is not a perfect geometric rectangular shaped object, so partitioning it based on grid is less realistic option. To avoid this problem geographic information of an area can be applied to define the boundaries where the area longitude latitude values are formatted as polygons. This represents each area boundary precisely and will be useful to solve this problem. We are utilizing the NYC taxi zones which is based on city planning’s neighborhood tabulation area and where each region is partitioned into polygon shape to its approximate neighborhoods. Instead of using grid based division, partitioning regions based on its actual geolocation information is more effective in terms of accuracy for demand calculation. Figure 3 plots the map using geolocations which represents the taxi zones of NY city and each region has a unique identification number. In our dataset each trip information for RHS contains pickup and destination location ID which comes from the polygons. We are calculating pickup-destination pairs according to their regions and create multiple one-dimensional arrays considering time period fit to be the input for our CNN. The raw data is processed and features are extracted. Then transformed into a matrix as the format of input data of the prediction framework. \( r \) is the number of inter area hire vehicle request from a pickup location \( i \) to destination \( j \) during a time period \( t \) which is represented by \( r_{ij}^t \). Similarly, for each pickup-destination pair, we will calculate the corresponding \( r \) at time \( t \). Finally, the total number of requests in the city \( R^t \) at period \( t \) which will consider each pair count and can be represented in form a matrix:

\[
R^t = \begin{bmatrix}
r_{11}^t & \cdots & r_{1n}^t \\
\vdots & \ddots & \vdots \\
r_{n1}^t & \cdots & r_{nn}^t 
\end{bmatrix}
\]

where \( n \) is the number of total area polygons or regions. The demand prediction problem aims to predict the demand at time interval \( t + k \), where \( t \) is current timestamp and \( k \) is the lag size. In addition to historical demand data, we also incorporated influential features such as time of day, day of week, holiday, weekend and areas of pickup and destination.

Following are the primary steps of our approach for demand prediction:

1. Extract pair-wise demand for region based pickup-destination locations.
2. Create 2D matrix for each time interval's RHS demand so that each matrix is a temporal snapshot of demand for all pairs.
3. Each temporal snapshot will be given input to 1D multivariate CNN model.
4. Compute feature information using CNN, store feature output in each subsequent array.
5. Use each subsequence with feature information from CNN as a new feature for biLSTM, memorize long-sequence features from each subsequence, use to predict the RHS demand for input pairs in future time interval.

B. One dimensional convolution network

A modified version of 2D CNNs called 1D Convolutional Neural Networks (1D CNNs) has recently been developed [3, 21]. Rather than matrix operations, forward and backward propagation in 1D CNNs require simple array operations. For this reason, the computational complexity of 1D CNNs is significantly lower than 2D CNNs. 1D CNNs with relatively shallow architectures (i.e., small number of hidden layers and neurons) are able to learn challenging tasks faster.

Main components of 1D CNN model used in this study are 1) convolution layers where both 1 dimension convolutions and sub sampling or pooling occur, and 2) fully connected layers that are indistinguishable to the layers of a typical Multi-layer Perceptron (MLP). CNN layer first performs a series of convolutions, the sum of this operation is passed through the activation function, \( f \), followed by the pooling operation. This is the main difference between 1D and 2D CNNs, where 1D arrays replace 2D matrices for both kernels and feature maps. As a next step, the CNN layers process the raw 1D data and learn to extract such features which are used in the prediction task performed by the MLP-layers. The detailed explanation about the forward and backward propagation functions can be found in [26].

C. Bidirectional LSTM

Bidirectional LSTM, or biLSTM, is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction and the other in a backwards direction. biLSTMs effectively increase the amount of information available to the network, improving the context available to the algorithm. The model we are using in the experiment consists of an output layer, a dense layer and a biLSTM layers. The input of the model requires a fixed-length vector. The dense layer is used to compress the dimension of the output vector of the biLSTM layer. The output layer collects the vector from the dense layer and outputs the desired regression value.

D. Proposed Model: CNN-biLSTM

The convolution layer in proposed model extracts the correlation between the input features and captures spatial dependency. Maximum pooling layer calculates the maximum value in each patch of each feature map. We flatten the output of the convolutional layers to create a single long feature vector. Next this tensor is passed through a bidirectional LSTM layer, which interprets the context from both directions. And it is connected to the final layer, which is known as dense layer. The network solves a regression task and final outcome of the model is number of demands for specific regions at different timestamps. Figure 5 illustrates the whole structure of the proposed model. As a desirable neural network to deal with the long-term dependencies in time series, LSTM is designed to overcome the vanishing gradients through a special gating mechanism [24]. LSTM avoids long term dependency problems by bringing the cell state, gate, and other schemes. The basic units of the LSTM network consist of three doors (input gate, output gate, and forget gate) and two memories (long-term memory and short-term memory). The input gate aims at selecting the needed new information and adds it to the cell state. The forget gate tends to remove the information that is no longer required by the memory cell, while the output gate decides what kind of necessary information in the cell should be output. Generally, the gating mechanisms can ensure the cells in LSTM network to store and update the essential information over long periods of time.

Figure 5 CNN-biLSTM architecture

Figure 6 An example of the temporal window structure
row starts by shifting the entire window by one hour which results in the prediction of the next hour. The diagram shows the prediction window starting from January 1, 12:00 AM leading up to predicting the demand for January 15, 12:00 AM horizontally and vertically shifting 1 hour from above row up to June 15 11:00 PM to predict demand for June 30 11:00 PM. The paper also implements some baseline models for comparison. A typical LSTM model and single one dimensional convolutional network consisting single CNN, max pool, flatten and dense layers. Another baseline model is combining CNN and LSTM models together.

V. RESULTS AND DISCUSSION

In order to compare our framework with the other baseline models, the experiments are conducted on a real dataset. No. of parameters for LSTM, CNN layers are 128, for dense layer 97. 32 filters and kernel size 3 is set for convolution layers. New layer has been added step by step with its previous layer, parameters are tuned and result is observed with new mode. In order to evaluate the predicted performance, the root mean square error (RMSE) and coefficient of determination, or $R^2$ are used. Root mean square error measures how much error there is between two data sets. The smaller an RMSE value, the closer predicted and observed values are:

$$Y_{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (X_{pred}(i) - X_{act}(i))^2}$$

(10)

The coefficient of determination, or $R^2$, is a measure that provides information about the goodness of fit of a model.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

(11)

Table 3 demonstrates the comparison among all models for 2018 NYC RHS dataset. Multiple pickup-destination pairs are compared and result for sample 5 pairs are shown. We considered same region based pair where pickup and destination ids are same and also different id based pairs. It is noted that CNN-biLSTM model outperforms the baseline models in terms of RMSE and $R^2$ for any pair. $R^2$ scores vary for different pairs but it is always closer to 1 which indicates our model works as expected and is a good fit. The table shows that for some pairs sequential model LSTM achieved better result than standalone CNN model and vice versa, however combining them both always showing a better prediction.

TABLE 3 RMSE AND $R^2$ VALUE COMPARISON AMONG ALL MODELS FOR 2018 NYC RHS DATA

<table>
<thead>
<tr>
<th>Regional pair</th>
<th>Model</th>
<th>$R^2$ value</th>
<th>RMSE value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pair 1</td>
<td>LSTM</td>
<td>0.717</td>
<td>76.414</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
<td>0.713</td>
<td>45.646</td>
</tr>
<tr>
<td></td>
<td>CNN-LSTM</td>
<td>0.718</td>
<td>33.837</td>
</tr>
<tr>
<td></td>
<td>CNN-biLSTM</td>
<td>0.720</td>
<td>33.705</td>
</tr>
<tr>
<td>Pair 2</td>
<td>LSTM</td>
<td>0.722</td>
<td>17.838</td>
</tr>
<tr>
<td></td>
<td>CNN</td>
<td>0.596</td>
<td>21.501</td>
</tr>
<tr>
<td></td>
<td>CNN-LSTM</td>
<td>0.714</td>
<td>18.088</td>
</tr>
<tr>
<td></td>
<td>CNN-biLSTM</td>
<td>0.731</td>
<td>17.542</td>
</tr>
<tr>
<td>Pair 3</td>
<td>LSTM</td>
<td>0.834</td>
<td>10.957</td>
</tr>
</tbody>
</table>

The proposed model CNN-biLSTM outperforms all models which indicates this model is able to capture spatial and temporal correlations successfully.

Figure 7 demonstrates the travel demand prediction curve for one day for a specific region pair in 2018. Prediction curves of all models are compared with the real value. From the illustration it is noted that CNN-LSTM and CNN-biLSTM models exhibit a trend similar to real data. Figure 8 depicts the percentage decrease in RMSE for out proposed model comparing to baseline models. This measurement is performed on 2018 RHS dataset. For pair 2 our model achieved the best accuracy which is 18.41% against the CNN model. Similarly, for this pair the proposed model achieved 22.65% improvement over the R2 value. Combining CNN and LSTM has achieved better performance than standalone CNN and LSTM models, applying bidirectional recurrent network (biLSTM) further improves the performance.

Among the state of the art models for ride hail service demand prediction, Liu et al. [22] achieved an RMSE of 19.85 in predicting only the origins and an RMSE of 1.32 in predicting demand between origin-destination pairs on NYC-TOD dataset. Chu et al. [23] achieved an RMSE of 1.015 in predicting taxi demand in NYC using a MultiConvLSTM. Shu et al. [21] applied CNN-LSTM model on historical data in Haikou (China) and obtained lower RMSE values than LSTM. The dataset applied in all these models is presented for grids and not neighborhood. On the other hand, the dataset used in our study is based on city neighborhood zones. There is no method to convert the data from grid format to neighborhood format, or vice versa, without making unrealistic assumptions and significantly lowering the data accuracy. Therefore, a
meaningful comparison between the accuracy of the grid-based versus neighborhood-based models would not be possible.

![Figure 8 Percentage-wise RMSE decrease (improvement) in the proposed model (CNN-biLSTM) versus LSTM, CNN and CNN-LSTM.](image)

VI. CONCLUSION AND FUTURE WORK

Travel demand modeling is an inherent part of smart transportation system. Forecasting travel demand can help us manage the hot spot of passenger demand in the next period, balance supply and demand and schedule vehicle resources for passengers. In this paper, a convolutional and recurrent network based deep learning model for ride hailing service demand prediction is proposed that takes advantage of both temporal and spatial properties on areal dataset. Proposed models’ performances are significantly beyond baseline models, confirming that it is better and more flexible for the travel demand prediction. In addition, we found that rectangular grid based partition method has several issues to calculate demand. Therefore, choosing a suitable partitioning method to predict travel demand is vital and has significant impact on increasing accuracies. In future, we will further improve the model and test with different pattern data sets. We want to extend the idea of utilizing the geojson information for partitioning method while demand calculation. For calculation, including data from adjacent cities can be used to verify model suitability.

REFERENCES


Transfer Learning-based City Similarity Measurement: A Case Study on Urban Hotel

Ganghua Zhang, Xiaoping Che, Shiyao Wei, Tao Na
School of Software Engineering
Beijing Jiaotong University
Beijing, China
19121730@bjtu.edu.cn, xpche@bjtu.edu.cn

Abstract—With the development of modern cities, multiple types and wide distribution of urban data has been gradually collected. Effectively using urban data to solve city development and planning issues has become a research hot-spot. Currently, the data scale in modern cities is quite different, and the fitting degree of machine learning algorithm based on single city is not mature yet. This paper studies the problem with transfer learning technique, and trains the prediction model of urban hotel development scale using multi-source city data. Based on the location data and related information of 15 different cities, the relevant knowledge is transferred, and a city feature extraction and similarity measurement framework is proposed.

Index Terms—Transfer Learning, City Similarity, Scale Prediction

I. INTRODUCTION

In order to solve the optimal location decision-making problem of enterprises, one of the most direct methods is to recommend the location of enterprise users. In recent years, a large number of academic papers have conducted in-depth research on the location recommendation method [1]–[3]. In the field of urban computing [4], the measurement of urban similarity [5] is one of the key issues. At the same time, the measurement of urban similarity is also the basis of urban transfer learning. At present, many methods of city similarity measurement have been explored. One of the dominant methods is the similarity measurement method based on the city feature vectors from different angles. For example, Sheng [10] proposed a geographic similarity region search model based on the approximate search algorithm based on quadtree. In their paper, K rectangular regions which are most similar to the target query area are found. The regional similarity evaluation method and feature extraction method proposed in their paper have important reference significance for the city similarity evaluation algorithm. Diserud [11] proposed a multi site similarity measurement method in the biological field. The measurement method uses the species information shared by more than two sites, and avoids the covariance problem between pairwise similarity in multi-site research.

II. RELATED WORK

A. City similarity measurement

At present, many methods of city similarity measurement have been explored. One of the dominant methods is the similarity measurement method based on the city feature vectors from different angles. For example, Sheng [10] proposed a geographic similarity region search model based on the approximate search algorithm based on quadtree. In their paper, K rectangular regions which are most similar to the target query area are found. The regional similarity evaluation method and feature extraction method proposed in their paper have important reference significance for the city similarity evaluation algorithm. Diserud [11] proposed a multi site similarity measurement method in the biological field. The measurement method uses the species information shared by more than two sites, and avoids the covariance problem between pairwise similarity in multi-site research.

B. Urban computing based on Transfer Learning

Based on the definition of transfer learning, many papers have studied the application of transfer learning technology in the field of urban computing. Wang discussed the general process of urban transfer learning in Smart City Development With Urban Transfer Learning [12], and provided guidance for urban planners and relevant practitioners on how to apply this novel learning method. This paper also puts forward the general steps to be followed in the application of transfer learning in urban planning, as well as case studies on public safety and traffic management, and summarizes some possible research directions.

Based on the urban transfer learning guide, a variety of transfer learning models for urban data have been proposed and optimized. For example, Yao [13] proposed multi-source transfer learning technologies. For example, Yao [13] proposed multi-source transfer learning frameworks of MultiSourceTrAdaBoost and TaskTrAdaBoost, which can effectively solve the problem of negative transfer [14] in urban data transfer learning, and can be applied to class identification and specific object detection of migrated objects. Tan proposed a multi-source transfer learning algorithm [15], which can make different views from different learning. The related research work can be summarized into three aspects: location recommendation algorithm, city similarity measurement and city measurement based on transfer learning.

DOI reference number: 10.18293/SEKE2021-011
sources and sources complement each other through collaborative training framework, so as to jointly utilize knowledge from different sources and views, and modify the distribution differences in different fields.

C. The shortcomings of existing research

There are some shortcomings in the two types of city similarity research directions introduced in this section. The first type of similarity research direction based on city feature vector still has the problems of optimizing the extraction method of feature vector and the selection of distance measurement method when measuring the city similarity. The problem of second type is that the standard of city score is not universal.

III. CITY SIMILARITY MEASUREMENT ALGORITHM

The similarity measurement between cities can be carried out for different data types in cities.

A. Urban similarity computing algorithm architecture

This section introduces the city similarity measurement algorithm proposed in this paper from the overall architecture level.

1) Basic concepts: Before introducing the architecture of city similarity algorithm, several basic concepts need to be defined clearly.
   a) City grid: In order to extract the features of cities and analyze the regional data better, we put forward the concept of urban gridding.

   The number of POI grid attributes can be defined as the variable shown in formula (1)

   \[ Prop^A_{g(i,j)} \]

   Where a represents a specific attribute and g(i, j) represents the coordinates of the lattice.
   b) City similarity: The city similarity measurement defined in this paper comes from the perspective of research objectives, which takes the feature dimensions to be studied and the related dimensions with strong correlation as the feature space of the experiment, which can solve the problem of difficult data acquisition and minimize the impact of redundant features on the experimental results.

2) Algorithm architecture: After defining the above basic concepts, the overall architecture of the city similarity measurement algorithm is shown in Fig.1.

   The similarity measurement part is the core part of the algorithm mainly measures the similarity between the macro feature vectors of two target cities. This algorithm uses Euclidean distance to calculate the distance between the two macro feature vectors, and takes the measurement results as a part of the final similarity results.

B. City similarity measurement

This section introduces the city similarity measurement algorithm in detail from the data preparation, feature extraction and algorithm construction, which is the basis and premise of building the transfer learning model introduced in the next Chapter.

<table>
<thead>
<tr>
<th>City</th>
<th>Beijing</th>
<th>Seoul</th>
<th>Tokyo</th>
<th>Shanghai</th>
<th>Dalian</th>
</tr>
</thead>
<tbody>
<tr>
<td>POI</td>
<td>635096</td>
<td>211089</td>
<td>391174</td>
<td>617870</td>
<td>119411</td>
</tr>
<tr>
<td>Grid</td>
<td>957</td>
<td>142</td>
<td>465</td>
<td>667</td>
<td>195</td>
</tr>
</tbody>
</table>

2) Feature extraction: The definition variable a represents the target prediction characteristics, the variable X represents the target research city, the variable R represents the grid row number of the city divided by the target research city x, and the variable C represents the column number of the city grid:

   \[ Num^X_{cells} = r \times c \]

   Equation (3) represents the total number of grids in the target study city X. Then there are:

   \[ Sum^X_A = \sum_{i=0,j=0}^{i=r,j=c} Prop^A_{g(i,j)} \]

   Equation represents the total number of feature a in the target city. After defining the basic variables that need to be used in the macro feature vector, the feature vector V of the macro part of city x is defined \( V^x_m \) define a S is the 19 feature dimensions of urban grid, where a is the target feature, B is the target feature S represents the rest of the features, and the
feature vector is defined as 18 dimensions, one of which can be defined as formula (4).

\[ f(\theta)_X = \frac{\sum_{i=0}^{c} P_{\text{IRF}}(i,j)*\text{Sum}_X^{\theta}}{\text{Num}_X^{\text{grids}}} \quad \theta \in [B, S] \]

(4)

3) City similarity measurement: The city similarity measurement algorithm proposed in this paper measures the similarity of the target city from two levels, namely, the whole city level and the single grid level. Among them, the overall level similarity measurement mainly uses the city macro feature vector extraction method introduced before. The measurement formula of the overall level similarity of city X and city Y is shown in formula (5).

\[ \text{Sim}_X^Y = \frac{1}{10000 * \sqrt{\sum_{i=0}^{17} (V_{X[i]} - V_{Y[i]})^2}} \]

(5)

The detailed measurement process is shown by the pseudo code of algorithm 1 below.

**Algorithm 1 GridSim \((R_q,T,k,M,m)\)**

**Require:** Query grid \(R_q\). The results show that the number of regions in the first cycle is \(M\); the number of main categories in the region is \(m\)

**Ensure:** CF-IRF feature similarity \((\text{Sim}_{CF-IRF})^T_{R_q}\); location similarity \((\text{Sim}_l)^T_{R_q}\);

1. \(CMI_i = \text{ExtractCategory}(T_i,m); CM = \text{ExtractCategory}(R_q,m)\)
2. Adjust\((T_i,CMI_i); \text{Adjust}(R_q,CMI); R_T = T_i\)
3. for \(T_i \in T\) do 4. \((\text{Sim}_{CF-IRF})^T_{R_q} = \text{avg}(\sum_i \text{Sim}_{CF-IRF}(R_q,T_i))\)
5. if \((\text{Sim}_{CF-IRF})^T_{R_q} > \min(R_T)\) then 6. \(R_T = R_T \cup T_i\)
7. end if 8. end for 9. \((\text{Sim}_{CF-IRF})^T_{R_q} = \text{avg}(\sum_i \text{Sim}_{CF-IRF}(R_q,T_i))\)
10. return \((\text{Sim}_{CF-IRF})^T_{R_q}\)
11. for \(R_j \in R_T\) do 12. \(\text{Sim}_l(R_q,R_j)\)
13. end for 14. \((\text{Sim}_l)^T_{R_q} = \text{avg}(\sum_i \text{Sim}_l(R_q,R_j))\)
15. return \((\text{Sim}_l)^T_{R_q}\)

**Algorithm 2 Cycle optimization algorithm**

**Require:** \(TRD0\) - Urban training data; \(Xd \in TRD0\)-characteristic distribution of each grid; \(Yd \in TRD0\)-label of each grid

**Ensure:** \(\theta_X\)-parameter
1. Initialization of network structure and Kaiming initialization
2. \(\text{epoch} \leftarrow 0\)
3. while \(\text{epoch} < \text{MAXEPOCH}\) do
4. The corresponding relationship between \(X_d, Y_d\) is calculated
5. ADAM
6. \(\text{epoch}++\)
7. end while
8. return \(\theta_X\)

IV. PREDICTION ALGORITHM OF URBAN HOTEL DEVELOPMENT SCALE BASED ON TRANSFER LEARNING

A. Prediction algorithm of urban hotel development scale

This section will introduce the flow and principle of the algorithm in detail from three parts: algorithm architecture, neural network structure and algorithm optimization process.

1) Algorithm architecture: In this paper, the urban hotel development trend prediction problem is defined as a classification problem. The category labels to be predicted represent the high, medium and low number of urban hotels in the target grid.

As shown in Fig.1 described before, the similarity between the target city and other cities is calculated. The city data with the highest similarity and the target city data are input into the neural network as the training set for iteration. Then, the normal Kaiming initialization method is used to set the initialization parameters of the neural network. The parameter optimization algorithm in the iteration process is Adam, and finally the model is compared on the test set. The performance of the model was evaluated. The training process of the model is shown in algorithm 2.

**B. Experimental results**

This section introduces the whole process of model training and the final experimental setup and results.

Beijing is selected as the target experimental city. Firstly, the similarity between the other 14 cities in the dataset and Beijing is calculated according to the city similarity measurement method described in Chapter 3. The distribution of the measurement results is shown in Fig.2(a).

The experimental results of target cities show that the higher the similarity with the target city, the better the learning effect is. Therefore, the city similarity measurement algorithm proposed in this paper can accurately measure the similarity degree of POI distribution between cities, and then transfer learning based on the similarity to get the prediction model of City Hotel development trend. The accuracy has been improved.

C. Algorithm visualization

Based on the prediction model of urban hotel development scale based on transfer learning, this section visualizes the established algorithm model and applies the algorithm model to practical application, which verifies the effectiveness of the model in practical application. The screen shots of running results are shown in Fig. 4.

V. SUMMARY AND PROSPECT

According to the given direction of the experiment, we design and implement the city POI feature extraction methods of macro level and grid level. With the distance measurement method of Euclidean distance and cosine similarity, we finally realize the city similarity measurement algorithm. Using this algorithm.

Based on the city similarity measurement algorithm, this paper introduces the transfer learning based deep neural network method, designs and implements the benchmark experiment.
At the same time, according to the results of city similarity distribution, six representative city data are selected to train the transfer learning model. The experimental results show that the urban data transfer learning effect and source. The results also verify the correctness and effectiveness of the proposed method. Finally, a visualization system of urban hotel development trend prediction algorithm is designed and implemented.

ACKNOWLEDGMENT

Thanks for the guidance of associate professor Xiaoping-Chen, School of software, Beijing Jiaotong University, and the help of Tao-Na and Shiyao-Wei in the experiment.

REFERENCES


A hierarchical RNN-based model for learning recommendation with session intent detection

Jinyang Liu  
School of Economics and Management  
Beihang University  
Beijing, China  
liujinyang@buaa.edu.cn

Chuantao Yin  
Sino-French Engineer School  
Beihang University  
Beijing, China  
chuantao.yin@buaa.edu.cn

Xiaoyan Zhang  
Sino-French Engineer School  
Beihang University  
Beijing, China  
xiaoyan.zhang@buaa.edu.cn

Kunyang Wang  
Sino-French Engineer School  
Beihang University  
Beijing, China  
wang_ky123@buaa.edu.cn

Hong Zhou  
School of Economics and Management  
Beihang University  
Beijing, China  
h_zhou@buaa.edu.cn

Abstract—Since the emergence of MOOCs (Massive Online Open Courses) in the last decade, online education continuously evolves. With the abundance of learning resources provided by MOOC platforms, recommender system can be used to personalize learners’ learning experience with respect to learning material consumption. To provide user-adaptive and beneficial recommendation result, the recommender system should be designed with respect to properties of the online learning context, especially the sequential property of learning behaviors. In this paper, we propose a novel model SOLR, a session-based sequential model for online learning material recommendation. We use a hierarchical RNN to model online learners’ learning sequences on both in-session and cross-session levels. Additionally, attention mechanism is used within sessions to model users’ learning session intent. The model is able to learn a hierarchical representation of users’ long-term learning history as well as short-term session sequential patterns. We conducted comparative experiments with session-based recommendation baseline methods as well as an ablation study on real-life MOOC dataset. The results show that our model achieves better recommendation results and provide justification for the sequential modeling and model training mechanism implemented in our model.

Keywords- online learning, MOOC, recommender system, RNN, smart learning

I. INTRODUCTION

Recent years have witnessed the rise of online learning. Since the emergence of MOOCs (Massive Online Open Courses) in the last decade, online education continuously evolves. One of the biggest online learning platforms Coursera had 37 million registered users and over 3,100 active courses in 2018 [2]. The abundance of learning resources including digital textbooks, exercises, video tutorials, on-site forums and blogs creates the room and necessity for user personalization and adaptation in online learning systems.

Recommender systems as a form of user personalization play an important role in online services, such as the field of e-commerce, online content consuming (video and music streaming, news, etc.) and social network. Recommendation techniques are usually classified into three categories: collaborative filtering, content-based and hybrid recommendation models [3]. Collaborative filtering studies the user-item interaction through user behavior or implicit feedbacks and predicts the user’s preference towards certain items. Content-based recommender uses item attributes and user profile as auxiliary information to match users with items. Hybrid techniques combine two or more strategies to meet specific requirements of the system [1]. Recently recommender systems using deep neural network gain much attention for their ability to incorporate multiple level of abstraction of the data using neural representation and deep structures and demonstrated great success in their performances [4][5].

There have been successful attempts applying state-of-art recommendation techniques for online learning systems. Collaborative filtering methods have been implemented in early learning management systems [6][7]. Content-based techniques combining with domain ontology such as case-based reasoning and attribute-based matching have been used successfully to recommend learning materials [8]. Although general approaches of recommender systems could be transplanted to online learning platforms, to achieve better performance there are unique challenges to take into consideration within the online learning context [9]:

1. Learning activities are organized in a sequential manner. There’s causality in learners’ learning histories, i.e. their background knowledge and materials they previously consumed. The sequential and progressive property is intrinsic within education which should be paid more attention to comparing to other recommendation scenarios such as e-commerce and streaming services.
2. A user’s learning path on the platform consists of several sessions. Within each session users demonstrate different learning behavior styles and strategies which should reflect on the materials recommended to them.

To tackle the problems stated above we propose a novel method using session-based sequential recommendation with hierarchical recurrent neural network to capture learners’ learning histories as well as their interest within sessions. In addition we use attention mechanism further enhance the session intent. Experiments show that our method achieves state-of-the-art performance in online learning dataset.

II. RELATED WORKS

The concept of sequential recommendation is primarily used in session-based online activities. For sequential recommendation it’s important to capture user’s long-term static preference as well as to predict user’s short-term behavior in order to recommend the immediate item for user’s need [10].

Traditional sequential recommendation techniques include sequential pattern mining, Markov Chains, sequential KNN and session-based matrix factorization. Sequential pattern mining derives from frequent pattern and association rule mining in that it mines a collection of ordered frequent patterns and then performs inference based on predefined support and confidence thresholds [11]. Markov chains (MCs) based methods postulate that the item a user consumes next depends on one or several items he consumed before that. Matrix factorization-based methods are among the most relevant and efficient methods nowadays. Twardowski [12] applied factorization machine to session-based recommendation. Rendle et al. [13] combined first order MC with factorization method and He et al. [14] used similarity matrix factorization combining high order MC and achieved desirable performance on sparse data.

There are several drawbacks in conventional methods. Sequential pattern mining methods suffer from scalability issues. Besides, they ignore the users’ differences in their behavior patterns thus lack the personalization we desire. KNN and matrix factorization have limited ability to capture sequential propriety across sessions. Markov Chain based methods fail to capture users’ long-term behavior tendency.

In recent years deep learning with neural networks has achieved great advancement in the field of natural language processing and computer vision. Deep neural models have the ability to incorporate heterogeneous inherent and contextual information of the input with low dimensional representations and reduce the effort of using hand-crafted features. Covington et al. [15] proposed a deep neural model for YouTube video recommendation. Cheng, Heng-Tze, et al. [16] proposed a deep-wide neural network structure recommendation framework. Both have achieved significant performance improvement comparing to conventional methods.

Deep neural models that have been adapted to sequential recommendation problems are mainly Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs). Hidasi et al. proposed GRU4Rec [17], using Gated Recurrent Unit to model the sequential item interactions and training the model using mini-batch parallel training. It’s the prototype for several later improved models [18][19]. By utilizing data augmentation strategy and improved pairwise loss function design the GRU4Rec models achieved better performance. The HRNN4Rec model developed by Quadrana et al. [20] first applied hierarchical RNN for recommendation to model users’ across-session and inner-session behaviors. Zhang et al. [21] further enriched the item side information by adding dwelling time in RNN recommendation scheme. Compared to RNNs, CNN structure does better in capturing global and non-consecutive sequential behavior with lower computational costs. Tang and Wang proposed Caser [22] using convolutional sequence embedding with horizontal and vertical convolutional filters to capture the point-level, union-level and skip-item behaviors.

Attention mechanism was first proposed in natural language processing for machine translation tasks [23]. It models machine’s “attention” by assigning different weights to parts of the input sequences which is ideal for modeling short-term intent in sequential recommendation problems. NARM [24] leverages attention mechanism for sessions in its encoder-decoder structure to model user’s purpose within sessions. Liu et al. [25] used attention mechanism and calculated the attention correlation between history and recent items.

III. METHODOLOGY

In this section, we first frame the session-based recommendation task for online learning. Then we describe our model including the model structure and training specifications.

A. Session-based Recommendation

A learner’s behavior on learning platform consists of several consecutive learning sessions which are loosely defined as the learning material sequences the learner consumes in a period of time. For example, a learner in one session beginning with logging in on the platform could revisit the tutorial video he watched during last session, begin a new video then go to the course forum for further exploration. To describe the problem formally, for a user u his whole lifetime activities on the platform could be represented as a set of sessions: $S_u = \{S_{u,1}, S_{u,2}, ..., S_{u,T}\}$. Each session consists of several learning materials the user consumes: $S_{u,m} = \{i_{m,1}, i_{m,2}, ..., i_{m,|S_m|}\}$. A session-based sequential recommender learns the user’s sequential behavior and learning history and tries to predict the most suitable material for the user to consume next within the current session.

![Figure 1. Illustration of the model structure](image-url)
B. Model

To tackle the specific problems of online learning recommendation task, we propose our model Session-based Online Learning Recommender (SOLR). Our model uses a hierarchical recurrent neural network structure with GRU to model users’ learning behavior during sessions and their learning history. Attention mechanism is leveraged in the session local encoder to capture users’ session intent. The model takes in item embeddings, hierarchically encodes users’ local and global sequential behavior. Within sessions the sequence representations are fed into a fully connected layer to produce the recommendation results. The complete model structure is shown in Fig. 1.

Embedding layer We use an attribute-aware embedding layer to produce the item representations taking into account the items’ category and type information. The category information could be the learning material’s subject and ontology type information. The information is the material’s form of its presentation (video, audio, forum thread, etc.). For an item i, the embedding layer produce the item’s representation:

\[ x_i = CONCAT(e_i^I, e_i^C, e_i^{tp}) \]  

(1)

Where \( e_i^I \), \( e_i^C \) and \( e_i^{tp} \) are the embedding of the item ID, category information and type information. The embedding matrices \( E^I \in \mathbb{R}^{D_i \times |I|} \), \( E^C \in \mathbb{R}^{D_c \times |C|} \) and \( E^{tp} \in \mathbb{R}^{D_{tp} \times |TP|} \) each transforms the item’s ID, category and type one-hot representation into embedding of dimensions \( D_i \), \( D_c \) and \( D_{tp} \).

RNN with GRU A Gated Recurrent Units is a more elaborated model first introduced to tackle the vanishing gradient problems in traditional RNN structure [26]. Using the mechanism of reset and update gates, GRU updates the hidden units in a selective and weighted manners in each step. Compared to GRUs, Long Short Term Memory (LSTM) networks are used more often in NLP tasks. However, [17] shows that replacing GRUs with LSTM led to worse performance in session-based recommendation. The activation of GRU is the interpolation between the previous activation and the candidate activation \( \tilde{h}_t \):

\[ h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t \]  

(2)

Where the update gate \( z_t \) is given by:

\[ z_t = \sigma(W_z x_t + U_z h_{t-1}) \]  

(3)

And the candidate activation \( \tilde{h}_t \) is given by:

\[ \tilde{h}_t = \tanh(W_x x_t + U(r_t \odot h_{t-1})) \]  

(4)

\( r_t \) is the reset gate and calculated in a similar manner as \( z_t \):

\[ r_t = \sigma(W_r x_t + U_r h_{t-1}) \]  

(5)

Matrices \( W_z \), \( U_z \), \( W_x \), \( U_r \), \( W_r \) and \( U_r \) are all parameters to learn during training. The whole updating process can be noted as: \( h_t = GRU(x_t, h_{t-1}) \) for simplicity.

Hierarchical RNN The paragraph above describes a classic RNN structure. However, in the context of session-based user behavior, the idea of the sequence of sessions differs from a long item sequence. Users’ learning behavior on learning platforms shows different inner-session and cross-session patterns. For example, sessions could be abstracted as learner’s taking one or several tutorials, following up with certain points of the lectures, taking exercises or going over all previous sections for refreshment. Each of those groups of activities within sessions differs with respect to the sessions’ purposes and behavior patterns. And the progression of sessions signifies the learners’ accumulation of knowledge and their achievement on learning paths. Hierarchical RNN was used in HRNN recommender [20] for session-based recommendation. The idea was to build above item sequences (i.e. sessions) RNNs another global RNN encoder to model the session evolution.

The session-level RNN takes within a session each item for input and predicts items the user is consuming next in this session. The user-level RNN takes each session’s last hidden state as input, and updates the hidden state for the next session’s initialization. We use upper corner \( s \) and \( u \) to differentiate the hidden state notations in the two levels. As illustrated in Fig. 1, the m-th session initialization is calculated as:

\[ h_{m,0} = \tanh(W_{init} h_{m-1} + b_{init}) \]  

(6)

Then within the m-th session, GRU propagates through each step:

\[ h_{m,t} = GRU_{sess}(x_t, h_{s,m,t-1}) \]  

(7)

After the last step’s update, the session hidden state is taken as input for the user-level update:

\[ h_{u,m} = GRU_{user}\left(h_{m,s}, h_{u,m-1}\right) \]  

(8)

Operating on two levels, hierarchical RNN models the inner-session dynamics as well as cross-session user evolution which can be justified with the real-life online learning experience.

![Attention mechanism](image)

Attention layer Attention mechanism is widely used in sequence modeling. During sequence prediction tasks, attention mechanism calculates each input’s contribution with respect to the current prediction. In our model during session propagation, to predict the current item we use all previously consumed items to produce a session intent embedding:
where the attention weights $\alpha_{t,j}$ are attained by using the query vector $w^q$ with softmax normalization:

$$\alpha_{t,j} = \frac{\exp(w^q h_{m,j}^T)}{\sum_{l=1}^{N_d} \exp(w^q h_{ml}^T)}$$  \hspace{1cm} (10)$$

As shown in Figure 2, the intent embeddings are fed to the fully connected layer to produce the final prediction for the next item.

C. Model training

We use user session parallel mini-batch training with prefix data augmentation method which is well established in the GRU4Rec family [17][18][19]. The user session parallel mini-batch training method groups sessions by user then sorts sessions within each group using time stamps. Groups of sessions are fed to the model and trained parallelly. If any of these user sessions end the next sessions are put in their places.

The model is trained with pairwise ranking loss function and negative sampling. The loss function we use is the TOP1-max loss, which is an improvement of the classic TOP1 function by focusing on the most highly ranked negative sample. Given a set of negative samples $N_d$, the TOP1 loss function is calculated as:

$$L_{TOP1} = \frac{1}{|N_d|} \sum_{j=1}^{N_d} \sigma(r_j - r_t) + \sigma(r_t^2)$$ \hspace{1cm} (11)$$

Where $r_t$ and $r_j$ are respectively the score of the target item and the score of the negative sample $j$. The second term in the sum represents a regularization by punishing the high score given to irrelevant items. The TOP1-max loss uses the softmax scores of all negative samples to weight each sample’s contribution:

$$L_{TOP1-max} = \frac{1}{|N_d|} \sum_{j=1}^{N_d} \text{softmax}(r_j) \cdot \left( \sigma(r_j - r_t) + \sigma(r_t^2) \right)$$ \hspace{1cm} (12)$$

The loss function focuses on the most wrongly rated items and alleviates the vanishing gradient problem when the number of samples increases [19].

![User parallel mini-batch training](image)

*Figure 3. User parallel mini-batch training*

We use the user parallel mini-batch training mechanism which is proposed by Quadrana et al. for the HRNN4Rec model. As shown in Figure 3, we group sessions by user then feed several user sequences parallelly to the model. In each iteration, the current mini-batch serves as input for the update of users’ $GRU_{user}$ and the next mini-batch serves as the ground truth for training. At end of the session, (8) is used for the update of the user’s $GRU_{user}$. If one user’s sequence ran out, a new user sequence will be put in its place with $GRU_{user}$ and $GRU_{session}$ reset.

For the negative sampling, the original GRU4Rec model as well as its derived methods including HRNN4Rec all use a mechanism called batch sampling to speed up the training process. Batch sampling takes other parallel sessions’ item in the same batch as the negative samples. However, in the case of sequential modeling with explicit user representation, this mechanism suffers from under sampling effect. Because each user sequence can only be exposed to limited parallel users during training. Thus, we opt for the commoner popularity-based sampling mechanism during training.

IV. EXPERIMENTS

The experiments led in this study consist in two parts: 1. To demonstrate the effectiveness of our model, we compared our model against five baseline methods for session-based recommendation on their performance in online learning material recommendation. 2. We realized an ablation study in which we deprived our model of its several functionality modules to show the effectiveness of the mechanisms we implemented in our model.

A. Dataset

We use the XuetangX dataset of user logs for our experiments. XuetangX is the largest MOOC platform in China which has provided over 1000 courses and has more than 10 million registered users. The XuetangX dataset contains users’ activity logs on the platform from August 2015 to August 2017. There are in total 698 instructor-paced mode (IPM) courses and 515 self-paced mode (SPM) courses. We use users’ activity logs for SPM courses since during SPM courses users have more autonomy with respect to learning behaviors. Table I shows the aggregative description for the dataset:

<table>
<thead>
<tr>
<th>Type</th>
<th>Total count</th>
</tr>
</thead>
<tbody>
<tr>
<td>logs</td>
<td>382,225,471</td>
</tr>
<tr>
<td>Forum activities</td>
<td>90,815</td>
</tr>
<tr>
<td>Web page activities</td>
<td>5,496,287</td>
</tr>
<tr>
<td>Assignment activities</td>
<td>3,139,558</td>
</tr>
<tr>
<td>Total materials count</td>
<td>1,227,078</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>enrollment</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>218,274</td>
</tr>
<tr>
<td>Users</td>
<td>123,719</td>
</tr>
<tr>
<td>SPM courses</td>
<td>515</td>
</tr>
</tbody>
</table>

B. Baseline methods

We compare our model against several baseline models:

- Item-KNN: Calculating the item-item cosine similarity based on the co-occurrence of items in sessions across
users. Regularization is applied to avoid high score. This method is session-based but non-sequential.

- **FOSSIL [14]**: A hybrid model fusing matrix factorization-based similarity model with high order Markov Chain to take into account both user long-term preference and short-term sequential pattern. In these experiments the maximum order of the Markov Chain is set to 2.

- **GRU4Rec [19]**: The improved version of GRU4Rec with prefix and dropout data augmentation mechanism and loss function with top-k gains. This model doesn’t have user representations thus lacks personalization.

- **HRNN4Rec [20]**: The hierarchical RNN model for session recommendation. It could be seen as our model without attention mechanism. Similar to the original model, the model is trained with user parallel mini-batch training. But we replace the original batch sampling mechanism with popularity-based negative sampling.

- **u-GRU4Rec**: A modified GRU4Rec model with recurrent user representation by joining sessions of the same user into a long sequence. Same to HRNN4Rec the model is trained with user parallel mini-batch training with popularity-based negative sampling.

**C. Experimental setups**

For our model, the item embedding, session hidden state and user hidden state are respectively of size 100, 256 and 256. For item embedding we include a category embedding using the course label to which the items belong. We train the model with user session parallel mini-batch training with batch size 500. The uniform negative sampling size is set to 1024.

For other neural network baseline methods, we use the same setup for the size of hidden state vectors and training parameters. For the two models with user representation (u-GRU4Rec and HRNN4Rec) we didn’t implement the batch sampling method used in their original methods for its negative effect of under sampling. Instead we used uniform sampling as used in our model with the same sampling size.

We use all users’ last session as test data and all their previous sessions as training data. The evaluation is also carried in a user parallel fashion.

**D. Results and analysis**

**General performance** As shown in Table II, our model outperforms the baseline models in recall and ranking evaluations. The following observations can be made:

Compared to the baseline methods’ performances on other recommendation datasets such as movie and e-commerce datasets in their original studies, the recall and ranking evaluations are rather high across models for learning material recommendation for XuetangX dataset. This can be interpreted as the unique behavior paradigm on online learning platforms. Users’ behavior sessions are more or less predefined by the way the materials are organized. When an online course is curated by a MOOC provider it’s usually segmented by several learning sessions i.e. lessons. Users usually follow the material sequences within the paths of the courses in most part of the learning activity. Most of the uncertainty happens when users drift away from the course pages and into the forum threads as well as the revisiting of former consumed materials across several sessions.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Recall</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HR@5</td>
<td>HR@10</td>
</tr>
<tr>
<td>Item-KNN</td>
<td>0.2581</td>
<td>0.3012</td>
</tr>
<tr>
<td>FOSSIL</td>
<td>0.3492</td>
<td>0.4607</td>
</tr>
<tr>
<td>GRU4Rec</td>
<td>0.3601</td>
<td>0.4613</td>
</tr>
<tr>
<td>u-GRU4Rec</td>
<td>0.4364</td>
<td>0.5370</td>
</tr>
<tr>
<td>HRNN4Rec</td>
<td>0.4388</td>
<td>0.5482</td>
</tr>
<tr>
<td>SOLR</td>
<td>0.4565</td>
<td>0.5521</td>
</tr>
</tbody>
</table>

Compared to non-sequential baseline model item-KNN, the sequential recommendation models have significant higher scores. This proves that online learning activities are inherently sequential thus justifies the methodology of using sequential recommendation techniques in the online learning domain.

Compared to non-neural methods (Item-KNN and FOSSIL), the three neural network methods have better performances on both ranking and recall evaluation metrics which proves that recurrent neural networks are better apt to sequential modeling.

Models with recurrent user representation (u-GRU4Rec, HRNN4Rec and our model) outperform GRU4Rec model which is a non-personalized session-based sequential method. This could be attributed to the differences of the sequential patterns with respect to personal learning style and preference in the learning behaviors among users. Models with user representation are able to incorporate these personalized sequential patterns into user-distinct recurrent states. Also HRNN4Rec and our model both have better performances than u-GRU4Rec which shows the benefit of the hierarchical recurrent user representation. The traditional RNN structure is proved to have difficulty in long sequence modeling. The HRNN structure uses cross-session level update to aggregate the sequential history and in this case is more fitting to the online learning scenario.

Overall our model outperformed all the baseline methods on both recall and ranking evaluation metrics. This could be attributed to the contribution of the user-level global representation and the contribution of attention mechanism within session.

**In-session performance analysis** We compared the four sequential recommendation models above with user representation (FOSSIL, u-GRU4Rec, HRNN4Rec and our model) in their session-level performances by breaking down the sequential session recommendation into three stages: begin-session, mid-session and the end-session. For begin-session performance we evaluate the recall and ranking ability of each model on the first two item of the session. The end-session is seen as the last item in the session and the rest is the mid-session.
From the results shown in Table III, we can see that our model and HRNN4Rec have better performance in beginning-session recommendation due to the session initialization using hierarchical user recurrent representation. With user representation the model can use the user’s learning history information and cross-session behavior to predict the beginning of the user’s next session. In mid-session and end-session recommendation our model has the best performance among the four, which is a testament to the session intent detection ability of the attention mechanism.

<table>
<thead>
<tr>
<th>Models</th>
<th>HR@5 begin</th>
<th>HR@5 mid</th>
<th>HR@5 end</th>
<th>MRR@5 begin</th>
<th>MRR@5 mid</th>
<th>MRR@5 end</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOSSIL</td>
<td>0.2041</td>
<td>0.3510</td>
<td>0.3587</td>
<td>0.1408</td>
<td>0.2782</td>
<td>0.2815</td>
</tr>
<tr>
<td>u-GRU4Rec</td>
<td>0.2603</td>
<td>0.4395</td>
<td>0.4497</td>
<td>0.1579</td>
<td>0.2996</td>
<td>0.3105</td>
</tr>
<tr>
<td>HRNN4Rec</td>
<td>0.3550</td>
<td>0.4406</td>
<td>0.4404</td>
<td>0.2071</td>
<td>0.3060</td>
<td>0.3230</td>
</tr>
<tr>
<td>SOLR</td>
<td>0.3556</td>
<td>0.4531</td>
<td>0.4587</td>
<td>0.2075</td>
<td>0.3141</td>
<td>0.3354</td>
</tr>
</tbody>
</table>

The effect of the length of user learning history
To further illustrate our model’s strength in users’ learning sequence modeling we inspected the performances of the neural network models with respect to the length of users’ learning history. The users were regrouped into three categories: users with short, medium and long learning history which contains respectively users with under 10, 10 to 30 and over 30 learning sessions. The models put in comparison are u-GRU4Rec, HRNN4Rec and our model, with the non-personalized GRU4Rec for the control group.

The results show that u-GRU4Rec with the traditional RNN structure has declined performance for users with longer learning history. In this case considering the average users in one sessions of the XuetangX user logs, the normal RNN structure has difficulty in modeling learning material sequences after 200 updates on average. However both HRNN4Rec and our model show the increase in performance with the growth in length of user history, with our model achieving higher performance especially in ranking metric.

We reason that the hierarchical RNN structure’s update mechanism is close to nature of user’s progression of their learning activities. Using HRNN the models have a better ability in modeling the accumulation and abstraction of learning history.

E. Ablation study
To demonstrate the effectiveness of our design of the model functionalities and choices for training mechanisms, we realized an ablation study where three mechanisms in the model are inspected for their contribution to recommendation performance: the attention mechanism, item embedding with extra-information and the popularity sampling mechanism. In each experiment we remove one of the mechanism and keep the other two then train and evaluate the model with the dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Ablation Study Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOLR</td>
<td>attention + extra embed + popularity</td>
</tr>
<tr>
<td>SOLR w/o attention</td>
<td>+ popularity</td>
</tr>
<tr>
<td>SOLR w/o popularity sampling</td>
<td>+ batch</td>
</tr>
<tr>
<td>SOLR w/o extra embedding</td>
<td>+ popularity</td>
</tr>
</tbody>
</table>

As shown in Table V, when the popularity-based sampling is changed to batch sampling used by the original GRU4Rec series of models, the model suffers from the most severe decline. This is due to the under sampling effect of using batch sampling method in user parallel mini-batch training. The model without the attention mechanism also has performance decline which is expected according to the comparison between our model and HRNN4Rec. However, the extra-information embedding with
course information and material type information doesn’t appear to have a big effect in recommendation performance. We reason that this is due to the particular behavior patterns of online learning with respect to course subject and material type. Usually the learning materials are organized by curators and educators such that users can follow through the predetermined learning path. And most users feel comfortable to do as such. Most of the randomness appearing in learning sequences is in session beginning and when users go to materials out of course structure such as wiki pages and forum threads. Thus most of the sequential information with respect to the learning materials’ course and type information is learnt during training with only the embedding of item IDs. Overall the experimentation results justified the mechanisms used in our model. The HRNN structure combined with in-session attention mechanism, extra-information embedding and popularity-based sampling mechanism has better ability in sequential recommendation and more adaptive to the online learning context.

V. CONCLUSION

This paper presented a hierarchical recurrent neural network model with attention mechanism for learning material recommendation on online learning platform. It captures users’ learning history and session behavior patterns in both global and session-level. Experiments conducted on real life data from XuetangX MOOC platform demonstrate the effectiveness of our model by comparing it to other sequential-based recommendation methods. The comparisons of the performances on session-level and with various user history lengths show that our model is capable to incorporate long-term user behaviors and to learn an aggregated representation of user history. By modeling users’ intent during a session using attention mechanism, the model can achieve better performance for recommendation within current session. The experiment results prove that our model has better compatibility with the highly sequential online learning behavioral context. We would further explore the potential of incorporating more material side information such as tag information and domain ontology which we believe can contribute to session-based recommendation for online learning. The model could also be applied in other domains of recommendation where the user behavior has an inherently sequential property.

ACKNOWLEDGEMENT

This work is supported by the National Natural Science Foundation of China (No. 61977003).

REFERENCES


Intelligent Preprocessing Selection for Pavement Crack Detection based on Deep Reinforcement Learning

Yan Lin  
Beijing University of Posts & Telecommunications  
Beijing, China  
linyan@bupt.edu.cn

Guosheng Xu  
Beijing University of Posts & Telecommunications  
Beijing, China  
guoshengxu@bupt.edu.cn

Siyi Li  
Beijing University of Posts & Telecommunications  
Beijing, China  
lsy_curry@qq.com

Guoai Xu  
Beijing University of Posts & Telecommunications  
Beijing, China  
xga@bupt.edu.cn

Jie Deng  
National Engineering Center of Road Maintenance Technologies  
Beijing, China  
dengjie@roadmaint.com

Jiankun Cao  
National Engineering Center of Road Maintenance Technologies  
Beijing, China  
caojiankun@roadmaint.com

Abstract—With the rapid increase of traffic, the pressure on road maintenance is gradually increasing. Pavement crack is a common problem in all kinds of pavement diseases. In the actual production process, pavement images have different kinds of noise influence. The proposed algorithm is to select optimal preprocessing methods for pavement images in various conditions to improve the accuracy of crack detection. The algorithm includes two parts, a crack detection network and an intelligent preprocessing decision system. The crack detection network identifies the cracks in road images. The intelligent preprocessing decision system selects the best preprocessing method for pavement images based on the deep reinforcement method. The experiment results indicate that the validity and effectiveness of our proposed method.

Index Terms—crack detection; deep reinforcement learning; intelligent preprocessing decision system.

I. INTRODUCTION

The highway transportation system is one of the most important large-scale public infrastructures. With the rapid growth of traffic, the pressure on road maintenance is gradually increasing. Pavement crack is a common problem in various pavement diseases. How to effectively improve the detection effect of pavement cracks is an urgent problem to be solved. With the continuous advancement of technology, automatic image acquisition and detection methods have gradually replaced manual detection. In the last few years, with the swift development of the identification techniques, Kim et al. [1] discuss the methods of detecting and recognizing road cracks in 2017. Iqbal et al. present that the image preprocessing is the basic step of image processing and computer vision. It includes basic operations, such as noise removal, cropping, brightness increase, and non-uniformity correction [2]. In the actual production process, there may be water stains, uneven illumination, shadows, and other interference information on the road surface. Different feature preprocessing methods are usually adopted for different road noise information. It is hard to shape a general system to detect pavement cracks in different conditions. In order to better identify the pavement cracks, we propose an algorithm that can autonomously select the optimal preprocessing method for road pavement images in various conditions. Cracks can be accurately detected when selecting the most suitable preprocessing method.

II. RELATED WORK

Deep reinforcement learning is based on reinforcement learning, supplemented by the strong generalization and feature extraction ability of depth models. Deep reinforcement learning has made great success in continuous decision-making problems, such as game playing [3] and machine control [4]. Reinforcement learning is the problem faced by an agent that learns behavior through trial-and-error interactions with a dynamic environment [5].

Deep reinforcement learning has four key elements: environment, state, action, and reward [6]. A reinforcement learning model can be built with these elements. The problem of reinforcement learning is to obtain an optimal policy for a specific problem, maximizing the reward obtained under this strategy. The policy is the sequential data that represents a series of actions [7]. Motivated by the advances and the features of deep reinforcement learning, we propose an intelligent preprocessing system for pavement crack detection based on deep reinforcement learning method. 1) The system can choose different preprocessing methods for pavement images in various conditions. 2) The system selects the optimal
preprocessing method for a certain image to improve the accuracy of the crack detection model.

The rest of the paper is structured in the following manner. In Section III, we describe our proposed system. In Section IV, we explain the details of the components of the system. In Section V, we demonstrate the experiments and the analysis of results. In Section VI, conclusions are provided.

III. SYSTEM OVERVIEW

A. Overall Structure

The general architecture of the proposed algorithm in this paper is illustrated in Fig. 1. The network is divided into two parts: the crack detection network and the intelligent preprocessing-decision system.

The goal of our algorithm is to raise the accuracy of the crack identification model by autonomously choosing a suitable preprocessing method for images. We utilize the deep reinforcement learning method to check the quality of the features extracted by the crack identification network and to judge the recognition of the crack identification model. If the accuracy of crack identification reaches a certain value, then the crack detection network can output the result. Otherwise, images will be transformed by the preprocessing method selected by the intelligent preprocessing-decision system.

The decision ability of the deep reinforcement learning method is crucial. It can fully understand the abstract features and make decisions on the basis of the operations of these features by integrating the functions of automatic feature extraction. We transform pictures with poor recognition results and re-value them by using the crack recognition model. Using the same parameters to raise the accuracy of the crack identification model is possible through the autonomous learning ability of the intelligent preprocessing-decision system.

B. Crack Detection Network

Crack detection network is responsible for the feature extraction, road image input, and crack recognition. The network is composed of several convolutional layers and five pooling layers alternately. These pooling layers can reduce the input images to their 1/32. Original 2200 × 3400 images are initially resized to 704 × 1088 and then reduced to 22 × 34 after going through five pooling layers.

The output of the crack detection network is a 22 × 34 matrix, each element of which represents if the corresponding block contains cracks. The value of 1.0 indicates that cracks may appear in this block. Such labeling is a one-to-one correspondence with the output of the neural network, which can efficiently conduct the end-to-end training of image labeling. Given the existence of multiple convolutional layers, the top-level neurons use the context information around its corresponding block to determine whether the small area is a crack, effectively utilizing the overall characteristics of images.

The output of the crack identification model and the target are matrices. Measuring the distance between each element in the two matrices is necessary. The number of blocks containing cracks is relatively small, and the number of blocks containing cracks in each picture can be rather different. We use positive sample dice coefficient (PSD) to measure the similarity of vectors as the objective function for training. The common formula for measuring vector similarity is defined as follows:

$$PSD = \frac{2|X \cap Y|}{|X| + |Y|},$$  \hspace{1cm} (1)$$

where $X$ represents the output matrix of the crack identification network, and $Y$ indicates the manually labeled matrix. $|X|$ and $|Y|$ represent the $L1$ norm of the two matrices. $X \cap Y$ represents the Hadamard product of $X$ and $Y$. When $X$ equals $Y$, $PSD = 1$; otherwise $PSD < 1$. The value of $PSD$ becomes large when the coincidence of $X$ and $Y$ becomes large. The value is only related to the proportion that the crack area is correctly identified and has no relation to the number of crack areas. Therefore, a common evaluation standard for different pictures exists. In practice, $X$ and $Y$ may be zero at the same time. To prevent the divide-by-zero error, we add a smoothing term to this formula:

$$PSD = \frac{2|X \cap Y| + \epsilon}{|X| + |Y| + \epsilon},$$  \hspace{1cm} (2)$$
C. Intelligent Preprocessing-decision System

In our paper, the intelligent preprocessing-decision system is a crucial part to improve the ability of the crack detection model. The main structure of this part is shown in Fig. 2. The system based on the deep reinforcement learning method judges the current result of the crack identification and transforms images with bad recognition results by using the selected preprocessing method. The transformed images are then placed in the crack identification network again, and a new round of calculation is performed.

The reinforcement learning method is used to select a suitable preprocessing method for the pavement images. To achieve this goal, four main components: environment, state, action, and reward should be defined.

Intuitively, the environment of the intelligent selection system is the feature space composed of the whole pavement image dataset. An intelligent system must learn which preprocessing method is beneficial for crack identification. It should transform images with bad recognition results by using the suitable preprocessing method.

We consider the feature space of images as the state. The initial state is the feature space of original images. After preprocessing images, the state is transformed into the next step, which is the feature space of the transformed images.

The intelligent system involves the preprocessing methods for pavement images, including Contrast-limited adaptive histogram equalization, bilateral filter, and morphological opening operator. We also add no operation on the images to the action set. Therefore, four actions are performed in total. In Section III, the reasons for selecting such preprocessing methods are explained in detail.

The system is trained to obtain the best strategy by the reward information received. The reward can show the performance of the identification algorithm and reflect whether the result of the crack recognition is better after a certain image preprocessing method. For each image, the accuracy of crack identification varies with different preprocessing methods. We regard the difference between the accuracy of the crack identification as the measurement of reward. If the accuracy of the transformed image becomes higher, then the intelligent preprocessing-decision system can receive a positive reward. By contrast, the negative reward can be received. That is,

\[
r_t = \begin{cases} 
r_a, & ACC_t > ACC_{t-1} 
-r_a, & ACC_t \leq ACC_{t-1} 
\end{cases}
\]

where \( ACC_t \) represents the accuracy of the crack identification of the current action, and \( ACC_{t-1} \) indicates the accuracy of the crack identification of the last action. \( r_a \) is a positive value to represent the positive reward, and \( -r_a \) is a negative value to represent the negative reward of the action. The structure of the intelligent preprocessing-decision system can be described in Table I.

<table>
<thead>
<tr>
<th>Algorithm of Intelligent Preprocessing-decision System</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Maximum training cycle ( I_{\text{max}} ), Maximum number of exploratory rounds ( E_{\text{max}} ). Maximum number of steps per round ( S_{\text{max}} ), update frequency of target network ( E_{\text{update}} ). Data amount of each batch ( M ), memory pool ( P ), and decay parameter ( \gamma ).</td>
</tr>
<tr>
<td><strong>Initialization:</strong> Random initialization of eval network ( Eval_{\text{net}} ) parameters ( \theta^Q ), and target network ( Target_{\text{net}} ) parameters ( \theta^Q \leftarrow \theta^Q ).</td>
</tr>
<tr>
<td>1. while the current training step (&lt; I_{\text{max}}):</td>
</tr>
<tr>
<td>2. while the current round step (&lt; E_{\text{max}}):</td>
</tr>
<tr>
<td>3. choose a picture from the training set;</td>
</tr>
<tr>
<td>4. obtain the feature matrix of the image as the initial state;</td>
</tr>
<tr>
<td>5. if the picture cannot be correctly read:</td>
</tr>
<tr>
<td>6. pass</td>
</tr>
<tr>
<td>7. while the number of current steps (&lt; S_{\text{max}}) and the episode has not finished:</td>
</tr>
<tr>
<td>8. select action ( a_t ), satisfying ( a_t = \arg \max Q(s_t, a, \theta) ) with probability ( 1 - \epsilon ) or randomly select action ( a_t ) with probability ( \epsilon );</td>
</tr>
<tr>
<td>9. transform the image by using the selected preprocessing method;</td>
</tr>
<tr>
<td>10. take the feature of the transformed image as the next state;</td>
</tr>
<tr>
<td>11. calculate the accuracy of the transformed image and obtain the reward;</td>
</tr>
<tr>
<td>12. store the memory ((s_t, a_t, r_t, s_{t+1})) into memory pool ( P );</td>
</tr>
<tr>
<td>13. randomly select data from ( P );</td>
</tr>
<tr>
<td>14. calculate the target value ( q_{\text{target}} = r_t + \gamma \times \max(\text{Target}<em>{\text{net}}(s</em>{t+1}, a_{t+1})) );</td>
</tr>
<tr>
<td>15. perform a gradient descent step on ( (q_{\text{target}} - q_{\text{eval}})^2 ) with respect to network parameters ( \theta^{Q'} );</td>
</tr>
<tr>
<td>16. every ( E_{\text{update}} ) steps reset ( \theta^{Q'} = \theta^Q );</td>
</tr>
<tr>
<td>17. end</td>
</tr>
<tr>
<td>18. end</td>
</tr>
</tbody>
</table>

IV. ACTIONS

The actions of the intelligent preprocessing-decision system are to transform the original image. Different preprocessing operations on original images are performed for such transformation. We choose four actions for crack detection and explain the reasons for selecting these operations. These operations include Contrast-limited adaptive histogram equalization, bilateral filtering, morphological opening operator, and direct output of original images (i.e., without preprocessing method), a total of four actions.

A. Contrast-limited Adaptive Histogram Equalization

Histogram counts the probability of each gray level appearing in images [8]. Histogram equalization (HE) utilizes the histogram to adjust the gray value of images for enhancing their global contrast and making their gray value evenly distributed in the histogram. The algorithm finds a gray value mapping:

\[
D_b = \frac{D_{\text{max}}}{A_0} \sum_{i=0}^{A_0} H_i,
\]
where $A_0$ represents the area of the image (the total number of pixels), $D_{\text{max}}$ indicates the maximum gray value of the original image, $D_a$ is the gray value of the original image, $D_b$ is the gray level of the converted image and $H_l$ is the number of i-level grayscale pixels.

Contrast-limited adaptive histogram equalization (CLAHE) [9] is a good complement to the shortcomings of HE. CLAHE [10] can preserve further details for images and minimize noise increase. Road images can have low or high overall brightness due to camera lighting. In this case, the contrast between cracks and surrounding road surfaces is reduced, and the characteristics of cracks are not obvious, which can cause missing recognition. The use of HE can effectively improve the contrast between cracks and surroundings, enhance the characteristic expressions of cracks, and make crack identification easy. CLAHE algorithm can reduce the noise interference enhanced by HE. CLAHE can also make processed images reduce noise and global contrast as much as possible while improving the local contrast and enhancing crack recognition.

We utilize CLAHE to avoid the problem of excessive brightness in certain areas of images and maximize the crack characteristics on the basis of enhancing the local contrast of images. The basic process is as follows:

1. Split images into $m \times m$ pieces.
2. HE of each block is carried out after the slope of the cumulative return function to avoid the increase of noise limiting the contrast.
3. Bilinear interpolation is used to eliminate the boundary between blocks.

**B. Bilateral Filter**

The purpose of an image filter is to eliminate noise interference while preserving the image edge information and contour as much as possible. Bilateral filter [11] is an edge-preserving filter based on the Gaussian filter, which only considers the distance between pixels, whereas bilateral filter considers the distance and gray value of pixels:

$$H(y) = \frac{1}{k(y)} \sum_{x \in S} p(x)G_d(x, y)G_r(p(x), p(y))$$

$$k(y) = \sum_{x \in S} G_d(x, y)G_r(p(x), p(y))$$

In Formulas (5) and (6), $G_d(x, y)$ represents the Gaussian weight of the distance between pixels $x$, $y$ and $G_r(p(x), p(y))$ denotes the gray value Gaussian weight between pixels $x$, $y$. Bilateral filtering can blur image noise by using different distances of pixels.

Asphalt-stirred stones generally repeatedly flatten Road surfaces; the largest difference between road surface images and other kinds of images is that many salt-and-pepper noises exist in road surface images. Such noises are determined by the inherent characteristics of roads. These salt-and-pepper noises may confuse the crack characteristics and cause the missing recognition or misidentification of the cracks. The bilateral filter can eliminate these noises to an extent and retain crack information.

**C. Morphological Opening Operator**

The two most basic operations in morphological transformation are erosion and dilation, and the convolution kernel sliding on original images is used to change pixel values. In the erosion operation, the center element maintains its original pixel value when all the pixels in the kernel are 255, otherwise, it becomes 0. This operation can erode the boundary of the foreground (white pixels). In the dilate operation, the pixel value of the center element is 255 as long as one pixel in the convolution kernel is 255. This operation can swell the foreground boundary, that is, etch away the background (black pixels). In mathematical morphology, using $\ominus$ for erosion and $\oplus$ for dilation is common. The opening operator can be described as follows:

$$A \circ B = (A \ominus B) \oplus B$$

Different sizes of the structural elements in the opening can result in various filtering effects, and the selection of different structural elements can lead to different segmentation, which means that different features are extracted. White noise is the main type of noise in road images. Therefore, the opening operation that can effectively remove white noise is a suitable preprocessing method for road images. White noise is removed after erosion, but black noise is increased, which can affect the accuracy of crack recognition. Images are then dilated to remove black noise, which can retain the crack characteristics. Therefore, the overall noise of images can be reduced, and the crack feature can be highlighted.

**V. EXPERIMENTS AND ANALYSIS**

The models used in the experiments are all based on Python and Pytorch. Pytorch is the python-version of the torch. A neural network is used in our algorithm, and a few parameters and computations are needed. Thus, we utilize GPU (Graphics Processing Unit) to accelerate the training process [12]. A server with Titan X Pascal GPU, which has a single-precision floating-point computing power of about 11 TFlops, accomplishes the experiments.

**A. Dataset Collection**

A dedicated digital camera with a vertical downward shooting angle takes the road pavement images. The vehicles with the camera must move along the road at a uniform speed in a straight line, taking a road pavement picture every two meters. The road pavement pictures are then numbered in sequence.

To fully reflect the algorithm performance, we collect approximately 10,000 HD road photos for training and testing. The road pavement dataset is composed of 8-bit single-channel gray-scale images. The image size is $2200 \times 3400$ pixels. The images are sliced into non-overlapping blocks, and each block includes $100 \times 100$ pixels. Therefore, 748 labeled blocks are found in each image. The blocks containing crack pixels are labeled as “1”, whereas those without obvious crack pixels
are labeled as “0”. A matrix $M$ containing only 0 and 1 can represent a road pavement image. A sample of the labeling image is illustrated in Fig. 3.

B. Evaluation

1) Comparison with Single Preprocessing Methods: The crack identification network is trained first. To evaluate our system, we identify the road cracks by using the crack identification algorithm with a specific preprocessing method and the one with the intelligent preprocessing-decision system. As mentioned in Section III, the preprocessing methods are CLAHE, bilateral filter, and opening. The intelligent system selects the most suitable preprocessing method for images. If the accuracy of the crack detection network improves, then the system can receive a positive reward. Otherwise, the system can receive a negative reward. The system finishes a training step when the accuracy reaches the point of 0.8, or when the last action is chosen as no operation.

The results rely on common definitions, that is, precision, recall, and F-1 score. We evaluate the prediction by calculating the PSD of the models. PSD emphasizes the positive examples in the prediction, which plays a critical role in our task.

The agent collects the total reward in an episode, and we periodically calculate the reward during the training process. We use 7,000 road images as the training dataset. Additional 3,000 pavement images are selected as the test dataset to evaluate network performance. Adam [13] is used as the optimization function. We use LeakyRelu [14] as the activation function to avoid the vanishing gradient problem. The learning rate of the system is set as 0.001. The greedy parameter of the greedy policy regularly changes during the training process. The reward function of the system is set as follows:

$$reward_0 = p_2 - p_1$$  \hspace{1cm} (8)

As described in Section II, $p_2$ represents the accuracy of the crack identification of the current action and $p_1$ indicates the accuracy of the crack identification of the last action. In real production, accurately identifying road cracks is crucial. The results in Table II show that the crack identification network with our intelligent preprocessing-decision system achieves higher precision and PSD. Therefore, our method performs better than the one with a single specific preprocessing method. Although the recall is slightly lower than CLAHE and opening, our proposed method achieves an overall F1 score of 78.36%, which is higher than the others. Such a score suggests that our system is helpful for crack identification to select the most suitable preprocessing method and improve the identification performance.

2) Comparison with different rewards: To demonstrate the importance of the reward function, we define the two other reward functions:

$$reward_1 = \begin{cases} 1, & p_2 > p_1 \\ 0, & p_2 = p_1 \\ -1, & p_2 < p_1 \end{cases} \hspace{1cm} (9)$$

$$reward_2 = \begin{cases} 1, & 0.8 > p_2 > p_1 \\ 2, & p_2 > 0.8 AND p_2 > p_1 \\ 0, & else \end{cases} \hspace{1cm} (10)$$

Table III shows that the reward function is crucial to system performance. $reward_2$ function performs better than the two other reward functions because the values of $reward_2$ function are always positive, which improves system learning.

3) Comparison with other methods: The last experiments reveal that our system achieves an improved performance with $reward_2$. We evaluate the performance of our system-$reward_2$ with several other existing methods. We select a few pavement images to show the crack detection results of different methods. The brighter color in Fig. 4 is detected as cracks by the corresponding algorithm. Table 4 shows the results of our system and the common edge and semantic segmentation methods: richer convolutional networks for semantic segmentation (RCF) [15], fully convolutional networks for semantic segmentation (FCN) [16], and DeepCrack [17]. Our system improves its performance in all aspects. As shown in Table IV, our system improves the performance by 17.2% relative to DeepCrack, second best, in terms of PSD.

VI. CONCLUSION

In the actual road detection work, the detected pavement images have kinds of background interference. Different pavement images adopt different preprocessing methods. This
paper proposed a method to select the optimal preprocessing method for pavement images in various conditions. The algorithm employs the deep learning method to form the strategy of independent selection of preprocessing method, so as to improve the detection effect of pavement cracks. The experiments and analysis show that our method performs better in the detection process which proves the validity and effectiveness of our method. And we are now planning for further expansion. In this study, we only select certain preprocessing methods. We believe that other preprocessing methods can be added to our system to improve its performance.

ACKNOWLEDGMENT

This work was supported by the National Key Research and Development Program of China (No. 2018YFB0803605).

REFERENCES


Inspect Defect of Power Equipment via Deep Learning Method

Qi Zhang¹, Qingsong Cai¹∗, Jianhui Zheng²

¹School of Computer, Beijing Technology and Business University
²School of Software, Tsinghua University
qizhang16@yeah.net, caiqs@btbu.edu.cn, zjh19@mails.tsinghua.edu.cn

Abstract- Efficient defect detection is crucial for maintaining a stable power system. One method is to check the status of electrical equipment through images or videos taken by UAVs (unmanned aerial vehicles), overcoming various difficulties of manual inspection in complex environments such as high-altitude, snowy or rainy days, etc. Unfortunately, it is still a difficult challenge to process the data returned by UAVs automatically and effectively. This study proposes a defect detection algorithm to process UAV images of power equipment based on Faster-RCNN. Additionally, we use multiple optimization strategies to improve the performance of the benchmark algorithm, including feature pyramid network, deformable convolution, online hard example mining, and data augmentation. Our results show that the average accuracy of experiments obtains nearly 60% improvement compared to that naive deep learning target detection algorithm. Meanwhile, this method is suitable for various defects of power equipment with high accuracy, thus benefiting thousands of households.

Keywords— power equipment; defect detection; UAV photography; deep learning

1 Introduction

Periodical inspection, troubleshooting, and timely repair are essential to the stability of the power system, especially to the long-distance high-voltage transmission circuit with complex topographical distributions. The common defects of electric devices are shown in Fig.1, including foreign matters in the tower, damage of the insulator, abrasion of wires, and corrosion of hardware. If these common electrical equipment defects are not found and fixed in time, it will cause extensive equipment damage and paralyze the power grid. To guarantee the steady operation of these circuit devices, unmanned aerial vehicle (UAV) photography has become the top choice for an electric system inspection. UAVs can capture images or videos of electric devices in a complex environment. In contrast, manual inspections often face tremendous challenges under which. And then, we can make out the defect type and position by processing these images or videos. Therefore, intelligent analysis of power equipment defects through images or videos obtained by UAVs plays a significant role in improving fault detection and repair efficiency.

Over the years, tremendous efforts have been made to process the images obtained by UAVs. In general, these methods can be roughly divided into three categories: The first and the most commonly used one is manual analysis [1], which is to inspect and analyze graphics or video information manually, bearing the highest accuracy. But, it also has obvious drawbacks. The high demand for the quantity and quality of professionals makes it expensive due to the vast amount of power infrastructure relying on manual inspection. Meanwhile, inspecting images manually for an extended period can cause fatigue and reduce judgment accuracy, thereby omitting some equipment defects and causing safety hazards. The second one is the traditional image-based defect detection method that inspects power lines by analyzing infrared images [2]. This method has satisfactory performance in detecting defects that lead to large temperature gradients, such as local high temperatures caused by wire corrosion. Nevertheless, not all power failures will cause temperature changes in power equipment, and infrared image analysis has its limitations. The third one is to use deep learning methods, which utilize data-driven training of convolutional neural networks to locate and classify defects [3]. Deep learning methods can automatically extract image features [4], simplify the image preprocessing process, and effectively improve target detection efficiency compared with traditional image fault detection methods.

However, the existing deep learning methods still have some shortcomings in solving these problems, such as insulator identification methods [5], or power transformer fault diagnosis [6], they can achieve good results under limited data sets. But these methods only have satisfactory performance for specific problems, which do not work for other kinds of defects. Because Faster-RCNN [7] is widely used in the industry and has high stability, strong versatility, which can relatively accurately identify multiple defects. This paper uses the Faster-RCNN algorithm as the benchmark algorithm. Then we propose an automatic detection method that can recognize
various defects in power equipment. Although Faster-RCNN is already a relatively mature detection algorithm, defect detection in power systems still faces many challenges: Firstly, there are so many types of defects to be detected that it is difficult for the algorithm to locate them accurately in UAV images. Secondly, complex and diverse backgrounds of pictures will bring annoying noise to the target detection algorithm, thus boosting the robustness of the algorithm in different scenarios must be taken into consideration. Thirdly, UAVs often shoot objects from different heights and angles, so the captured images’ target has different sizes and angles. The recognition algorithm needs to be able to detect targets at multiple scales.

This paper proposes a defect detection method for power equipment aerial images based on the improved Faster-RCNN algorithm. The system framework is shown in Fig.2: UAV aerial photography obtains videos or images of power equipment in different environments and transmits these images to the server after relevant processing. The improved Faster-RCNN target detection algorithm can automatically detect defects in power equipment images and automatically classify them, and then professionals will troubleshoot and repair the defect.

According to the characteristics of the UAV’s power equipment images, the improvements made to the original Faster-RCNN algorithm in this paper are as follows: First, to solve the problem of different resolutions and variables camera distances of images, we add a multi-dimensional feature extraction module to adjust the network structure, combining top-down and bottom-up feature extraction methods to detect defects of different sizes in images of different scales. This improvement allows the algorithm to obtain a larger receptive field when paying attention to minor electrical equipment defects. Second, because the defective shapes of power equipment are primarily irregular, the traditional convolution shape is relatively single and does not have strong adaptability to defect detection of power equipment. We use deformable convolution to enhance the feature extraction of defects of different shapes. Third, we use online hard example mining (OHEM), and data augmentation methods to solve the problem that there are generally only a single or a small number of electrical equipment defects in a single image captured by UAV. The improved algorithm framework pays attention to positive samples in the training process. It has a greater response to the area where the power equipment is located and where the power equipment is defective. More details are shown in Section III.

Our major innovations and contributions are as follows:

- This paper proposes an image defect detection method for power equipment based on deep learning. The method realizes integrated intelligent processing of defect detection, including UAV shooting, detecting and classifying defects automatically, and repairing electrical equipment.
- We add modules to optimize the performance of the native fault detection algorithm based on Faster-RCNN, including multi-scale feature extraction, deformable convolution, online hard example mining, and data amplification.
- Through the 5847 images of UAV’s electrical equipment with different resolutions for comparison experiments, it is proved that our algorithm for detecting defects of electrical equipment improves the accuracy by 60% on average compared with the benchmark algorithm. The method reduces the problem of excessive resource consumption in the maintenance of basic power facilities, and it is of great significance to national life and economic development.

This paper is organized as follows. In Section II we discuss some related work. In Section III, We will present our main idea. In Section IV we will show our experiment results, and we conclude in Section V.

## 2 Related Work

Although UAVs can capture numerous images and videos for electric equipment, they can not detect the defects of the equipment by automatically surveying those data. Therefore, image post-processing plays a critical role in defect detection after UVA image acquisition. The conventional processing method relies on the experience of the workers, which includes processing the images manually and marking the potential danger information via visual inspection. The cost of this approach is relatively high due to its experience dependence, labor enrichment, and material resource dependence. Thus, there is an unmet need to develop novel approaches to detect defects with better efficiency and lower cost.

Infrared image-based processing is one of the most commonly used methods for fault detection. Kazuo Yamamoto et al.[12] detected the transmission lines’ faults in power equipment by fusing infrared images with RGB images, which enlarged the difference between those two images and contributed to targeting extraction. Based on this research, Walter Fetter Lages et al.[13] developed a real-time fault detection module for transmission lines. Although this method simultaneously provides and visualizes the faults information in power lines, it can only detect transmission lines with temperature difference characteristics. Moreover, the detection ability for some other types of faults is weak. Additionally, the operating ambient temperature of UAVs is relatively complicated. During the shooting process, the imaging attitude is uncertain, and there are many interferences. Furthermore, different infrared devices cannot work together, which limits the accuracy of the detection.

With the development of modern digital image technology, artificial intelligence, and other disciplines, diverse methodologies have been developed to optimize image post-processing and achieve varying results[14]. Since traditional image processing methods above-mentioned cannot address the unmet need in fault detection for power equipment, many scholars turn their attention to deep learning methods to explore novel detection methods with better efficiency. With the breakthrough of artificial neural network theory, an artificial neural network has gradually become an accurate and efficient solution to learn specific tasks and complete the corresponding tasks.
The literature [15] applied the AlexNet as a feature extraction network when using the random forest algorithm to detect insulator failures. Also, for detecting the fault in insulators, The literature [16] introduced multi-layer perceptrons to extract the location-related features and local contour features of the captured images. Moreover, The literature [17] adopts the neural network method to filter out the background noise in detecting the transmission line. Notably, with the combination of convolutional neural network and wavelet transform, The literature [5] can detect faulty transmission lines more accurately due to the better classification and extraction of fault features. However, the deep learning method has made progress in detecting electrical equipment with drone aerial photography, especially in accuracy and automation. The types of faults detected by related studies are relatively single. Most of them are in the preliminary exploration stage. Unlike the above deep learning methods, we will describe how to solve the problem more universally by improving the benchmark algorithm in the next part.

3 The proposed method

This section will introduce how to apply deep learning methods to identify and classify defects in images. This paper aims to realize intelligent processing as much as possible. Moreover, the Faster-RCNN algorithm is widely used in aerial inspection [18] [15] and has high stability [19]. Therefore, we select the Faster-RCNN as the benchmark framework for defect detection in UAV images of power equipment. However, defects are small targets, and the traditional convolutional layers decrease sensitivity when deepening in the actual scene, it is easy to ignore small targets. So we select the residual network [20] Resnet101 as the feature extraction network. As the overall accuracy of the benchmark algorithm is still not high, we consider the specific problems encountered in UAV photography to make three improvements to achieve higher accuracy in the following part.

3.1 Feature pyramid network

The detected objects in the images have various scales due to the different shooting distances and defect sizes. So the algorithm needs to detect targets of various scales.

Traditional deep feature extraction networks in the shallow layer can extract more detailed features that contain rich information. They also have a more incredible response to small-scale targets. Nevertheless, contextual semantic information misses due to the lack of receptive fields. The network continues to downsample while the feature extraction network is deepening. This way increases the receptive field corresponding to each area in the feature map. The features extracted at this time contain rich semantic information. However, the feature extraction networks have a more excellent response to large-scale targets while many details in the image are lost simultaneously. This shortcoming is fatal for defects detection of power equipment because power equipment defects are generally small targets. Suppose we can extract multi-dimensional feature information and can achieve multi-scale feature extraction. In that case, the accuracy of detecting defects in electrical equipment captured by UAV will be further improved. So this paper implements a feature extraction process by extracting multi-scale features from the bottom-up and fusing the pyramid structure of different stages from the top-down to achieve the first stage optimization.

Figure 3: Schematic diagram of improvement

Figure 4: The process of multi-scale feature extraction using feature pyramid network, which integrates the top-down and bottom-down feature extraction methods

Fig. 4 shows the process of feature pyramid network. Firstly, the feature maps obtained in each stage are fixed as 1x1 convolution, then the number of channels is fixed as 256. Secondly, the lower-layer feature map is up-sampled by two times so that the feature map of the upper layer after convolution can scale the same size as the feature map of the lower layer. Then the bottom layer feature map fuses with the up-sampled upper layer feature through convolution and addition so that the high-level semantic information is merged with the detailed low-level information. Finally, The networks use 3x3 convolution to convolve the added feature maps to eliminate aliasing effects. Fig.3(a) shows the framework with the feature pyramid network.

3.2 Deformable convolution

The convolutional neural network proposes candidate regions and generates candidate frames by autonomously learning after the Faster-RCNN algorithm extracts image features. The corrosion of hardware is a common type of electrical equipment defect with irregular shapes. Additionally, The convolution kernel of traditional convolutional neural networks is generally rectangular or square. So it is a great challenge to detect the defects of the conductive line using the convolution with a regular structure. A fixed-shaped convolution kernel can sample a fixed-shaped area on the entire image, but the area covered by its receptive field is fixed for a given convolution kernel, which is not a good design. Moreover, The function of the high-level convolutional layer is to encode low-level features
and extract high-level features. However, targets at different locations often have different shapes and scales, limiting the coverage of the receptive field area, thereby restricting the detection of targets of different shapes. In the end, we adopt deformable convolution like Fig. 5 to accept defects. The convolution adjusts the scale and the coverage of the receptive field adaptively through learning, which will improve detection accuracy.

As shown in Fig. 3(b), this paper uses deformable convolution in the last three stages of ResNet101, enabling the feature extraction network to convolve appropriate locations at different scales. It improves the accuracy of small target detection.

3.3 Online hard example mining and data augmentation

There are often few or even no defects in the images of electrical equipment, which means that most of the candidate frames are based on the background. Thus, it is essential to correctly distinguish foreground and background. Correctly dividing the categories of positive samples for defect classification is also essential. This paper uses online hard example mining and data augmentation to improve the response-ability of the detection framework to targets.

Online hard example mining

In Faster-RCNN, the candidate region generation networks generate many candidate regions, many of which are negative sample boxes that do not intersect or intersect with the target box to be detected. These samples are easy to train samples. It is easy to reduce the loss function of the network by identifying these candidate sample frames as background. The ratio of the intersection with the foreground target frame (that is, an area ratio, which comes from the two candidate frames generated in the feature extraction stage, and divides their intersecting part and their merged part) of the difficult to train negative sample is relatively large. However, it does not exceed the set positive sample threshold. Therefore, it is necessary to focus on training these negative sample frames with a higher intersection ratio to be detected.

Online hard example mining can find difficult to train samples during the training process and increase weight of these samples to the loss function. The traditional method of regional candidate network to propose regional candidate frames is to randomly select 512 candidate frames to calculate the loss function according to the ratio of foreground and background 1:3 after the regression network refines and classifies the candidate frames. Instead, the online hard example mining method first calculates the loss of all the candidate boxes after refinement and classification, sorts the loss from high to low, and selects a total of 512 positive samples in a ratio of 1:3, and uses these samples to train network. Furthermore, it can improve the training effect by using the trained samples to train the network again. Then, such a method will make the sample’s relative characteristics more evident than the traditional method.

Data augmentation

Data flipping. UAVs often capture images of electrical equipment from different directions, but the defects of the electrical equipment are invariant. It means the defect information will still exist after flipping an image vertically or horizontally. Therefore, during the training process, we randomly flip the images and the device defect frame marked by the image, and the data volume finally increases. Simultaneously, it improves the robustness of the detector by using these modified images to train the detector and allows the detector to learn useful information from images in different directions and angles.

Data scaling. UAVs usually do not maintain the same height in the process of photographing electronic equipment, and they cannot guarantee the same route and height every time. Furthermore, the resolution of UAVs of different brands is also quite different. Therefore, the images are randomly scaled to simulate the real environment in the training process of the model. Then the model will have a greater response to power equipment defects with different scales. Fig.3(c) shows that the detection framework with online hard example mining and data augmentation is effective in classification and regression.
After making relevant improvements, the next part of the experiment will prove that the improved detection framework has achieved obvious results.

4 Experiments

In this part, the traditional Faster-RCNN algorithm will be used as the basic framework to experimentally verify the ability of the improved system framework to detect image defects of power equipment.

4.1 Dataset

This experiment’s data set consists of 5847 images of power equipment captured by UAVs, collected by one of the largest power companies in China. There are several types of electrical equipment defects in these images, such as foreign matters in the tower, damage of the insulator, abrasion of wires, and corrosion of hardware, and their resolutions are different. The data is divided into the training set, the verification set, and the test set at a ratio of 8:1:1. We verify the validation set when the training phase is completed (all training data are trained once) in training the model. We save the best model on the validation set and apply it to the test set to record the final result.

4.2 Evaluation indicators

Precision and Recall are important indicators for model evaluation. A comprehensive evaluation of recall rate and precision will be meaningful for evaluating defect detection of power equipment. Simultaneously, we also add the Average Precision and the mean Average Precision as measurement indicators according to the accuracy requirements of power system defect detection. The Average Precision refers to the average value of the maximum precision under the condition of calculating different recall rates, shown in Eq.(3):

\[
AP = \int_0^1 p(r) \, dr
\]

where \(AP\) stands for the Average Precision, and \(r\) refers to the recall rate. \(p(r)\) represents the highest precision when the recall rate is \(r\). The mean Average Precision refers to the average accuracy of each category, shown in Eq.(4):

\[
mAP = \frac{AP}{C}
\]

where \(mAP\) is the average accuracy of multiple types, and \(C\) is the total number of categories of the target to be detected. It should be noted that when calculating \(AP\) and \(mAP\), we use the prediction frame whose intersection ratio with the real target object’s outer frame is more significant than 0.5 as the average accuracy calculated by the positive sample.

4.3 Comparative experiments

This paper sets up four stages of comparative experiments to study whether the improvement of Faster-RCNN is useful. Faster-RCNN with Resnet101 as the feature extraction network is set as the first group of comparative experiments recorded as experiment 1. Because the improvement of the benchmark system framework is divided into three parts, three sets of experiments are set to compare with experiment 1 during the investigation. Experiment 2 is the result of adding multi-scale feature extraction to the benchmark framework. Experiment 3 is the result of adding variable convolution based on experiment 2, and experiment 4 is the result of adding online difficult sample mining and data augmentation based on experiment 3. The intuitive comparison of detection accuracy can reflect the effectiveness of the improved target detection framework for power equipment images.

4.4 Experimental results

As shown in Tab. 1, experiment 2 increases the average accuracy of experiment 1 by 19.3% after adding multi-scale feature extraction. According to our analysis, the reason is that the resolution of images captured by UAVs is relatively large. Images entered into the network are often reduced by a certain percentage. The feature extraction network also downsamples the images by 32 times, which will cause the inherently relatively small defects to be significantly reduced or even lost during the down-sampling process. These small defect features can be well preserved and detected on the high-resolution feature map after adding multi-scale feature extraction.

<table>
<thead>
<tr>
<th>BEN</th>
<th>FPN</th>
<th>DC</th>
<th>OHEM/DA</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>√</td>
<td></td>
<td></td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>√</td>
<td>√</td>
<td></td>
<td>62.3</td>
</tr>
<tr>
<td>3</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>68.1</td>
</tr>
<tr>
<td>4</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>69.3</td>
</tr>
</tbody>
</table>

Table 1: The impact of algorithm improvement at each stage on the mean Average Precision. In particular, BEN stands for the benchmark algorithm, FPN stands for feature pyramid network, DC stands for deformable convolution, OHEM/DA stands for online hard example mining and data augmentation.

Additionally, the average accuracy of experiment 3 increased by 5.8% after using deformable convolution because experiment 3 is more adaptable to the shape of power equipment after adding deformable convolution. Finally, the average accuracy of experiment 4 increased by 1.2% by adding online hard example mining. We analyze that the model becomes easier to converge during the training process and learns from more difficult samples to get a more robust classification and detection capabilities. So experiment 4 further improves the average accuracy.

Fig. 7 shows the performance of different framework improvements in different categories of defects. In the original benchmark algorithm, this category can hardly be detected since the abnormal wire is small. With the addition of multi-scale feature extraction, the defects of this category are significantly increased. Because foreign matters in the tower are large and easy to detect, the benchmark target detection framework already achieve high detection accuracy, there are still some small-scale targets in this category of defects because the shooting distance of UAVs is relatively long, so multi-scale feature extraction can still improve the detection accuracy. De-
formable convolution can further improve accuracy when it detects frayed wires, damaged insulators, and foreign matters in the tower of different shapes. Finally, the algorithm is easy to jump out of the local minimum and converge better after adding online hard example mining.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BA</td>
<td>3.8</td>
<td>63.7</td>
<td>64.7</td>
<td>39.9</td>
<td>43.2</td>
</tr>
<tr>
<td>OA</td>
<td>50.2</td>
<td>85.9</td>
<td>73.6</td>
<td>68.2</td>
<td>69.5</td>
</tr>
<tr>
<td>IR</td>
<td>12.210</td>
<td>0.349</td>
<td>0.138</td>
<td>0.709</td>
<td>0.609</td>
</tr>
</tbody>
</table>

Table 2: The accuracy improvement ratio of the improved optimal algorithm and the benchmark algorithm. 1, 2, 3, 4 respectively represent abrasion of wires, foreign matters in the tower, damage of the insulator, and corrosion of hardware. BA, OA, IR respectively represent the benchmark algorithm, optimal algorithm, increase ratio. The results show that our algorithm exceeds the benchmark, which indicates the effectiveness of our approach.

Between comparison of the results of experiment 4 and experiment 1, it can be seen that the benchmark target detection framework is a better solution to the problem that minor defects in high-resolution power equipment images can hardly be detected. For foreign matters in the tower, damaged insulators, and large-scale clamp corrosion, the three major types of defects’ accuracy are increased by about 30%, 15%, and 70% respectively. As shown in Tab 2, the average accuracy of each category has increased by about 60% eventually. However, the experimental results show that the accuracy of this method is still low for small-sized objects such as wires, so the results left much to be improved. Therefore, we can further try to use multi-task training methods to detect small-sized objects in a targeted manner while retaining the existing algorithm framework. Some results of our experiment are shown in Fig. 8.

![Image of defect detection](image)

Figure 8: Visual display of defect detection: Tower refers to foreign objects in the tower, cable refers to abnormal conductors, insulator refers to damaged insulators, and rust refers to metal corrosion

5 Conclusion

In this study, we propose a detect defect detection algorithm based on improved Faster-RCNN to process images captured by UAVs. We use a multi-scale feature extraction method to detect targets of different scales on images of different resolutions with high accuracy. Meanwhile, we find that traditional convolution cannot adapt flexibly due to the shapes of power equipment are multilateral. We introduce deformable convolution to solve this problem. Moreover, we improve the performance of defect recognition by studying the attention degree of the object to be detected in the labeled data. In general, this research has improved the accuracy of intelligent detection of defects in power equipment. Also, it has dramatically reduced the consumption of human and material resources in the maintenance of basic power facilities, which has a very high application value.

References


[20] Q Chen, B Yan, R Ye, and XJ Zhou. Insulator detection and recognition of ex-

469
A method for generating various style Chinese fonts in the absence of training data

1st Kang Shi  
School of Computer Science and Software Engineering  
East China Normal University  
Shanghai, PR China  
51184501142@stu.ecnu.edu.cn

2nd Tian-Ming Bu*  
Shanghai Key Laboratory of Trustworthy Computing  
East China Normal University  
Shanghai, PR China  
tmbu@sei.ecnu.edu.cn

Abstract—In recent years, the generation of arbitrary style fonts has drawn broad attention. At present, the common idea of most methods is to train the model through a large number of text samples of a specific style, so that the model can learn the font style, and then automatically generate all the text of the style. However, in the case of insufficient sample data, almost all the current methods fail. In this paper, we investigate how to generate arbitrary style Chinese fonts automatically, especially when the sample data of specific style is insufficient. In order to solve the problem, we propose a new frame. The method can be used to generate the fonts of Chinese ancient calligraphers whose relics are rare. The experimental results show the effectiveness of the new frame.

Index Terms—text style transfer, generative adversarial network, convolutional neural network

I. INTRODUCTION

The text style transfer is an end-to-end image conversion, which is to generate a new image combining the content of the source image and the style of the target style image. In the case that there are enough samples of the target style, after seeing different characters of the same style font, the generative model can learn the target style, and then generate all the characters of the style. But how to automatically generate all the characters in the target style when there are not enough samples, is a problem that has not been solved yet.

In order to solve the problem, this paper proposes a new framework, which is composed of two parts: the first part is a model which is called the GTPD model, used to get the probability distribution of the font type of training sample, and the second part is the text style transfer model. The GTPD model rebuild training data set based on the font type probability distribution of a small number of samples of a specific style. The training data set constructed by the GTPD model is then used for the training of the text style transfer model. Further, because the existing text style transfer models perform not very well, we improve the existing style transfer models and design a multi-style transfer model with better performance. The structure of the new framework is shown in Fig. 1. In summary, the main contributions of this paper are in two aspects:

- we optimize the existing text style transfer model and propose a multi-style transfer model. The new model can not only generate more realistic text, but also learn multiple different styles of fonts at the same time. In addition, the new model can also generate the same font of traditional Chinese characters by learning the style of simplified Chinese characters.
- we propose a new framework for text style transfer in the absence of training data.

II. RELATED WORK

Gatys et al. [1] successfully applied the Convolutional Neural Networks (CNNs) [2] to neural style transfer, breaking the bottleneck that one program can only transfer one style.

When the CNNs were applied to neural style transfer, some researchers attempted to apply the CNNs to text style transfer. Yunchen Tian et al. [3] established an open source project on Github and proposed the Rewrite model. But the performance was not good. Pengyuan Lyu et al. [4], based on the Auto-Encoder model, proposed the Auto-Encoder Guided GAN (AEGAN) model for generating Chinese calligraphy characters. Compared AEGAN with pix2pix, the loss function of AEGAN model contains the reconstruct loss function, which is aim to guide the transfer to learn the detailed stroke information from auto-encoder’s low level features. Samaneh Azadi et al. [5] proposed the multi-content GAN model, and tried to learn the style through a few Latin characters with a specific style, and then generated all the Latin characters of that style. Shuai Yang et al. [6] proposed a scale-controllable module to empower a single network to continuously characterize the multi-scale shape features of the style image and transfer these features to the target text. In Anna Zhu’s paper [7], the output text font essentially is the same as the input, except the decoration added on the output text font. While in our work, the input and output are different fonts. We solve the problem of style transfer of
different fonts from only a few referenced samples(output font).

III. PROPOSED METHOD

In this paper, we improve the existing text style transfer model in some aspects. Firstly, based on the pix2pix model, constant loss function and category loss function are introduced to make the generative model perform better in the generated details. Then, a new framework is proposed to solve the problem of style transfer cannot work in the absence of training samples. The multi-style transfer model and the new framework are described in detail below.

A. Constant Loss

The learning process of human beings is the process of analogy. Our generative model is actually to learn how to map the sample space to the target space, namely, learn the mapping from a input domain to an output domain. Inspired by this, we apply the constant loss function proposed by Yaniv Taigman et al. [8] to our generative model.

Given two related domains $S$ and $T$, the generative model needs not only to learn to map from domain $S$ to domain $T$, but also to ensure the correlation between $S$ and $T$ during the mapping process. Assume that element $x \in S$ and the generative model is $G$. Since having the same feature is the premise of analogy learning, we hope that the feature of $x$ should have a correlation to the feature of $G(x) \in T$. In this case, we want to find a multivariate function $f$, which makes $f(x) = f(G(x))$.

To solve the above problems, we rebuild our generative model structure. As shown in Fig. 1, the reconstructed generative model $G$ is composed of two parts, one is a multivariate function $f$, and the other is a generative function $g$.

After applying multivariate function $f$ to generative model, we hope $\|f(x) - f(G(x))\|$ is as small as possible. Thus we have an additional component of the loss function of generative model $G$:

$$L_{constant} = E_{z \sim D} d(f(z), f(G(z)))$$

B. Category Loss

Before applying category loss function to our model, the model once can only learn one target style. Inspired by AC-GAN [9], we apply category loss function to our model, so that we can learn several font styles in one training and realize multi-style transfer model of text. To let the generative model learn multiple font styles at the same time, each font corresponds to a category label. The discriminative model not only determines the authenticity of the input sample, but also introduces another classifier to determine the category of the input sample. The loss function of the discrimination is composed of two parts: the log-likelihood of the correct source $L_S$, and the log-likelihood of the correct class $L_C$:

$$L_S = E \left[ \log P(S = \text{real} \mid X_{\text{real}}) \right] + E \left[ \log P(S = \text{fake} \mid X_{\text{fake}}) \right]$$
target style. When the sample data of the target style is insufficient, by using the GTPD model, we firstly get the probability distribution of the font type based on a small amount of sample data. Then we reconstruct the training data set according to the probability distribution of the font type. Next, the GTPD model will be described in detail below.

The GTPD model is essentially a multi-classification model. Unlike the classification task, we don’t want the GTPD model to tell us the type of font that the input belongs to. In fact, after the GTPD model is trained, the input to the GTPD model often does not belong to any font type in the sample space. Through the probability distribution of font type got by the GTPD model, we can see the relationship between target font and fonts in our sample space to a certain extent, that the output is the probability distribution. The formula is as follows:

\[ y = \frac{1}{M} \sum_{i=1}^{M} (p_1, p_2, \ldots, p_N), \]

where \( y \) represents the average probability distribution corresponding to \( M \) images input, and \( N \) represents the number of font types contained in the GTPD model’s sample space. In general, after getting \( y \), we take the font types of the three largest probability values to build the training data set. The data set constructed contains the above three fonts, and the quantity ratio is equal to the probability ratio in the probability distribution. Namely, \( m : n : q = p_x : p_y : p_z \), where \( m, n \) and \( q \) are the numbers of characters of font \( x \), \( y \), and \( z \) in the constructed training data respectively, and \( p_x, p_y, p_z \) are the probability of font \( x \), \( y \), and \( z \) respectively.

We use a deep neural network to construct the GTPD model. As is known to all, the structure of deep neural network directly affects the performance of the model. We measure the quality of GTPD model by its classification accuracy on the test set.

E. Implementation Details

1) Multivariate Function \( f \). The generative model of CGAN consists of an encoder and a decoder. The network structure of the generative model includes an encoder and a decoder. We use encoder to fit multivariate function \( f \).
Encoder is used to do feature extraction, which is very suitable for fitting multivariate function $f$.

2) GTPD Model. The loss function of the GTPD model is cross entropy function. In the case of insufficient sample data, we use the existing fonts in the GTPD model sample space to reconstruct the data set as input to the style transfer model. The GTPD model sample space should contain font types as much as possible that have similar texture to the target style font.

IV. EXPERIMENTS

In this section, we complete a large number of comparative experiments. Firstly, we verify the improvement of the generative model consists of constant loss and category loss function. Then, we demonstrate the effectiveness of the multi-style transfer model through the comparison experiments of different models. Finally, in the absence of samples of the target style, we carried out a lot of experiments with the new frame and used it to imitate the works of ancient calligraphers.

A. Data Set

We collect 10 kinds of common fonts and some Chinese calligrapher fonts as our training data set. In our experiments, we all use font Song (宋体) as the standard font.

B. Constant Loss and Category Loss

We take font Song as the source font (content) and font Yan (颜体) as the target style font (style), and randomly select 1500 simplified characters from these two fonts to construct a paired data set. The training set and the verification set are constructed at a ratio of 9:1 respectively. The pix2pix model, the model with constant loss function only and the model with constant loss function and category loss function introduced are respectively trained using the constructed dataset. As shown in Fig. 2, introducing constant loss function and category loss function to our model leads to better generation effect.

C. Comparison with Baseline Methods

In this subsection, we compare our method with the following baselines for text style transfer.

- Rewrite [3]: Rewrite is a simple top-down Convolution network with big convolution kernel size and stride. The network is minimized by L1 loss and total variation loss.
- Pix2pix [10]: Pix2pix is a conditional GAN based image translation network, which adopts the skip connection to connect encoder and decoder. Pix2pix is optimized by L1 distance loss and adversarial loss.
- Auto-encoder guided GAN (AEGAN) [4]: AEGAN consists of two encoder-decoder networks, one for image transfer and another acting as an autoencoder to guide the transfer to learn detailed stroke information.

For comparison, we take font Song as the source font (content) and font Kai (楷体) as the target style font (style), and randomly select 1500 simplified characters from these two fonts to construct a paired data set. The results are shown in Fig. 3.

The multi-style transfer model can generate the font of traditional Chinese characters by learning style from simplified Chinese characters. We randomly select 1500 simplified characters from different font libraries as training data. After training the model, we let the model to generate these types of traditional characters. The generated results are shown in Fig. 4. The simplified characters are already exist in the font library, and we generate the fonts of traditional characters corresponding to these simplified characters.

![Fig. 4. The generation results of traditional Chinese characters by learning style from simplified Chinese characters.](image)

D. New Framework

In the case of insufficient samples of the target style, we use the new framework to achieve text style transfer, and generate arbitrary characters of the target style. Firstly, we should train the GTPD model. We randomly select 2500 words from 13 common fonts (方正兰亭超细黑简体、方正黑体、方正楷体、方正仿宋简体、黑体、中易楷体、楷书、幼圆字体、华文楷体、华文宋体、华文行楷、方正兰亭、柳格简体) as the training data of the GTPD model. The output of a GTPD model is a vector of 13 dimensions, each of which corresponds to a font type. A value per dimension represents the probability that the input character belongs to that font type.

After training the GTPD model, we randomly select several characters from Suiliang Chu’s calligraphy work 《雁塔圣教序》 as input to the GTPD model (font Suiliang Chu is not one of the sample space fonts). Based on the output of GTPD model, we reconstruct the training set of the text style transfer model. Then we generate the whole calligraphy work based on the constructed training set. As shown in Fig. 5, the left is the characters of Suiliang Chu generated by using the new frame, the right comes from the network, which is the real work of Suiliang Chu.

V. EVALUATION AND DISCUSSIONS

We evaluate our proposed method as well as other baselines on the database we collected.

1) Effect of the constant loss function and category loss function: Fig. 2 shows the improvement of the model by introducing constant loss. It can be seen from the results that, after constant loss and category loss is introduced, the generated characters are more similar to the target...
characters in details. After further observation, it can be found that there are some polluted pixel points in the image generated by the pix2pix model. After introducing the constant loss function, these polluted pixel points disappear. After introducing category loss function, the model can not only learn a variety of different font styles at the same time, but also improve the generation effect.

2) Effect of the muti-style text style transfer: Multi-style text transfer model is the text style transfer model realized by introducing constant loss function and category loss function on the basis of the pix2pix model and adjusting the structure of the pix2pix model. As shown in Fig. 5, the multi-style transfer model has a good performance in the text style transfer.

3) Effect of the new framework: The new framework is used to solve the problem that the text style transfer models cannot work when the target style sample is insufficient. We propose the GTPD model to analyze the style characteristics by a small number of samples, and then reconstruct the data set for training the multi-style transfer model. It can be seen from Fig. 5, that the characters generated by using the new framework are very close to real characters.

Fig. 5. The text generated by the new frame and real work.

VI. Conclusion and Future Work

In this paper, firstly, we propose a multi-style text transfer model with good performance. The new model can not only generate more realistic text, but also learn multiple different styles of fonts at the same time. In addition, the new model can also generate all traditional Chinese characters by learning the style of simplified Chinese characters. And then, we propose a new framework to solve the problem that the text style cannot work when the target style sample is insufficient. The new framework is composed of GTPD model and multi-style transfer model. The experimental results verify the effect of the new framework.

In fact, everyone’s writing has a unique style, all belongs to a kind of font. Experiments in this paper show that each font corresponds to a different probability distribution of font type. So how to identity the handwritings through the probability distribution of font type is one of the future work.

REFERENCES

Multi-Fusion with Attention Mechanism for 3D Object Detection

Ning Wang, Ping Sun*
School of Software Engineering
Tongji University
Shanghai, China
*Corresponding author email: pingsun@tongji.edu.cn

Abstract—Artificial intelligence gradually plays the essential role in automatic driving, such as 3d object detection. Many state-of-the-art 3d detection frameworks fuse point cloud data and image data to perceive the surrounding environment of the vehicle. However, these approaches focus more on vehicle detections, and for objects with less point cloud sampling, such as pedestrians and cyclists, the performance is moderate. In this paper, we propose the multi-fusion framework with two kinds of attention mechanisms to solve the above problem and improve the detection accuracy of 3d objects. The proposed 3d attention mechanism with voxel sparse information is utilized in the framework. This framework contains two important modules: point fusion with 2d attention and voxel fusion with 3d attention. These modules firstly obtain the image features by projecting the lidar point or 8 vertices of the voxel to image feature maps. Then, these modules perform attentive fusion on the voxelized image features, point-wise image features and lidar data. Our evaluation on the challenging KITTI dataset, including 3d and bird's eye view metrics, demonstrates great improvements, especially at objects with less point cloud sampling.

Keywords-3d object detection; multi-sensor fusion; attention mechanism; convolutional neural network

I. INTRODUCTION

With the rapid development of artificial intelligence, great breakthrough has been made in the automatic driving. 3d object detection is an essential task in the automatic driving. Compared with 2d object detection, 3d object detection can obtain richer information such as the depth, position and volume, which helps to better perceive the surrounding environment of the vehicle. Lidar is the most used sensor for 3d object detection. Many early researches detect 3d objects from lidar point cloud [1, 2]. However, single sensor has its own disadvantages. For example, lidar cannot obtain intuitive image information. In this work, we focus on the multi-sensor data fusion for 3d object detection. On the basis of lidar point cloud data, the fusion of image data is helpful to give full play to the advantages of each sensor and improve the perception of multiple environments.

A. Challenges

3d detection algorithms only driven by lidar suffer from the loss of texture information and the sparsity of point clouds. Missing texture information causes many false detections between objects of similar size. Very sparse point clouds of small or distant objects lead to missed detections. To address these challenges, recent researches augment lidar point clouds with image features and learn to fuse features. Some researches [3, 4] utilize image features to generate 2d proposals, and then extract 3d features from the lidar points related to these 2d proposals. These approaches rely too much on reliable 2d detection results. In these methods, for the undetected object in the image, even if it has obvious features in the point clouds, it is difficult to detect it. Many algorithms [5, 6] project point clouds onto image features and then perform feature fusion. However, these approaches have high dependency on the reliability of high-resolution lidar point clouds and perform poorly when the lidar points are not sampled.

B. Our Contribution

To deal with the above problems, the approach that reduces the reliance on high-resolution lidar point clouds, and increases the weight of image features when the lidar points are extremely sparse is expected. In this paper, we propose the multi-fusion framework with two kinds of attention mechanisms to achieve the above expectations. The proposed approach extends the recent algorithm Multimodal VoxelNet (MVX-Net) [6]. Specifically, this proposed framework contains two important modules: point fusion with 2d attention and voxel fusion with 3d attention. These modules obtain the image features by projecting the lidar point or vertices of the voxel to image feature maps. The combination of these two modules not only ensures the accurate association between the image features and the point clouds, but also reduces the dependence on the high-resolution lidar point clouds.

General attention mechanisms distribute attention according to image features and can’t be directly applied to voxelized features. Inspired by the 2d attention mechanism, we propose the 3d attention mechanism for lidar point clouds. This mechanism takes dynamic voxelized data as the inputs, applies sparse 3d convolutions and produces a 3-dimensional spatial weight, which contributes to the selection of the effective voxelized features. What’s more, considering that the sparsity information is weak before the attentive fusion, we apply the sparsity feature to voxelized image features.

The main contributions can be summarized as follows:

- The multi-fusion framework performs attentive fusion on voxelized image features, point-wise image features and lidar data. This framework preserves the detailed

DOI reference number: 10.18293/SEKE2021-115
image features without overly relying on the effectiveness of the lidar point clouds.

- 3d attention mechanism is proposed for lidar point clouds, which contributes to distributing the attention to voxelized features. What’s more, the sparsity of lidar point clouds is utilized to enrich the voxelized features before the attention mechanism.

- Experiments on KITTI dataset demonstrates that our framework better handles error prone cases, and effectively reduces false detections caused by similar shape of point clouds, especially for objects with less point cloud sampling.

II. RELATED WORK

A. 3D Object Detection From Multi Sensors

The multimodal 3d object detection fuses multi-sensor data, such as the LIDAR and RGB data. The realization of the multimodal fusion relies on the synchronization of multiple sensors in time and the transformation of spatial coordinates.

Two-Stage Algorithm: These algorithms can be divided into three categories: based on multiple views, based on 2d proposals and based on semantic segmentation methods.

Multiple Views Method: Chen et al. [7] proposed the MV3D algorithm, which firstly generates the 3d proposals by the lidar data and projects them to the bird’s eye view, the front view and the image view. Then multi-view feature fusions are performed to refine the proposals. Later, many researches perform the multi-view fusion of different sensor data based on the 3d region proposals.

2D Proposals Method: Qi et al. [3] developed F-PointNets. This algorithm generates a 3d area for each 2d proposal, and applies PointNet++ [8] to obtain the point cloud features in the area. Zhao et al. [4] proposed the Point-SENet module to predict the scale factor and integrated the PointSIFT module to predict the direction.

Semantic Segmentation Method: In these methods, the existing semantic segmentation algorithm is used to eliminate most of the background points, and high-quality proposals are generated on the foreground points. Yang et al. [9] developed IPOD to remove most of the background points. Vora et al. [5] proposed the PointPainting, which appends the semantic features and the semantic prediction scores to the point cloud features. The accuracy performance is improved, but the inference speed is very slow.

The main disadvantage of Two-Stage Algorithms is that the two-stage operation slows the inference and training speeds, and requires higher computing resources for the computer. What’s more, Multiple Views Method firstly generates 3d proposals, and then utilizes image information to refine the proposals. This causes these algorithms to rely heavily on 3d proposals generated from point clouds only. Meanwhile, 2D Proposals Method pays more attention to reliable 2d detection results and weakens the effect of 3d point clouds. Therefore, an algorithm that can balance multimodal feature weights and complement each other is expected.

One-Stage Algorithm:

Sindagi et al. [6] proposed MVX-Net. This algorithm projects the non-empty voxels generated by VoxelNet [2] into the image, and uses a pre-trained network to obtain image features for each projected voxel feature. Then the combination of these image features and voxel features generates 3d detections. Though, the above method reduces the dependency on the availability of lidar points, the voxel projection reduces the accuracy of image features. In MVX-Net, authors also presented the point fusion. However, this method can’t reasonably select effective image features from high dimensions and has poor performance in low point cloud sampling.

B. Attention Mechanism

Attention plays an important role in human perception. Human vision obtains key areas by quickly browsing the whole picture, and then devotes more attention resources to the key area to obtain more detailed information, while suppressing other useless information. The attention mechanism in deep learning draws on the human attention and is widely used in various types of deep learning tasks such as natural language processing, image translation [10] and network pruning [11].

Recently, several researches have applied the attention mechanism to convolutional neural networks (CNN). Wang et al. [12] proposed Residual Attention Network, which stacks attention modules to generate attention-aware features. Hu et al. [13] proposed SENet which is generated by SE block. This architecture focuses on the channel relationship and uses global average pooling features to compute channel-wise attention. S Woo et al. [14] presented Convolutional Block Attention Module (CBAM). This module exploits both spatial-wise and channel-wise attentions and then the attention maps are multiplied to the feature map for adaptive refinement.

However, these attention mechanisms operate on 2d convolutions and cannot be directly applied to 3d voxel operations of point clouds. In addition, the difference between the voxel and the pixel is that voxels have different densities, and the application of the previous attention mechanism will lack the consideration of the density of voxels.

III. MULTI-FUSION FRAMEWORK WITH ATTENTION MECHANISM

We present a multi-fusion framework with two kinds of attention mechanisms to fuse the RGB and point cloud features. Inspired by MVX-Net, the presented framework contains two important modules: point fusion with 2d attention and voxel fusion with 3d attention. These modules firstly obtain the corresponding image features by projecting the lidar point or vertices of the voxel to image feature maps. Then, these modules perform the attentive fusion on the lidar data and the image features. The proposed 3d attention mechanism for lidar point clouds takes dynamic voxelized data as inputs and applies the sparse 3d convolution, which helps to generate the effective voxelized features. What’s more, the sparsity distribution of voxels is exploited for the attention mechanism, which enriches the image features with the sparse information.

The overall architecture is illustrated in Fig. 1. First, we utilize the 2d convolutional neural network which takes RGB
images as inputs and extracts multi-level image features. Next, voxel features are encoded from lidar point clouds and two attentive fusion modules are performed to generate fused features. Then, the 3d backbone network takes the concatenated features as inputs, and the head network outputs the 3d detection results.

A. Image Feature extraction

Residual Network (ResNet) [15] is made up from residual blocks with skip connections, which effectively increase the depth of network and the ability to extract features. Balancing the computing resource and model performance, we eventually adopt ResNet with 50 layers (ResNet50) as our image backbone.

Feature Pyramid Network (FPN) [16] is a feature extractor that combines multiple resolution features via a top-down pathway and lateral connections, which enriches the outputs with multi-dimensional information. We use FPN as the image neck network.

Given RGB images, the image backbone network generates multi-scale features. Then, these feature maps are merged by element-wise addition in the image neck network, which finally outputs several sets of image features with rich semantics.

B. Voxel Feature Encoding

Voxel feature encoding (VFE) is a voxel feature learning network from VoxelNet [2]. The input of VFE is the point cloud data after the dynamic voxelization, which records the coordinates of the voxel where the point cloud is located and the raw features of the point cloud. The VFE network first obtains point-wise features through FCN learning, and then utilizes max pooling to generate the locally aggregated features. These features are regarded as the voxel global features, which are concatenate to each point-wise feature.

Stacks of such VFE layers transform low-dimensional point cloud features into high-dimensional voxel features, which will be the input of the Voxel Fusion with 3d Attention module. In order to obtain the input of the Point Fusion with 2d Attention module, the voxel features are discretized into the point cloud and connected with the initial point cloud feature.

C. Point Fusion with 2d Attention

This module associates lidar point clouds to image features and perform the attentive fusion to obtain the point-wise features with additional image features. We adopt point fusion strategy for the accurate association information, which is described in MVX-Net [6]. Moreover, this module applies 2d attention to make fused features more expressive.

The details of this module are illustrated in Fig. 2. Given the multi-scale image features produced by the image backbone and point features produced by voxel feature encoding, this module outputs the attentive fusion features. In details, firstly 5 sets of 256-dimensional image features at different scales are input into the module. Then the point-wise image features are calculated:

$$I_{pw} = BL(-1 + 2 \times \frac{M(T', P, C_{coor})}{[w, h]}, I)$$

$T$ denotes the transformation matrix, $P$ denotes preprocessing parameters, $C_{coor}$ denotes the 3d point cloud coordinates and $(w, h)$ is the width and height of the image. $M$ represents the coordinate transformation function and $BL$ represents the bilinear interpolation function. $I$ denotes the initial image features and $I_{pw}$ denotes the point-wise image features.

The 2d channel attention is performed in the above discrete point features, which is inspired by CBAM [14]. In details, as shown in Fig. 2, we respectively calculate the channel average feature and channel maximum feature of 640 dimensions, and use the sigmoid operation to obtain the 640-dimensional weight vector. The same attention mechanism is applied to the point-wise features obtained by VFE to generate the 64-dimensional weight vector. Then the fusion feature is generated by the concatenation of two attentive features.

The main advantage of this module is that for point cloud features with both raw features and voxel features, the attention mechanism can amplify effective features. In addition, the final
fusion directly performs on the raw point clouds, and this point-to-point mapping effectively reduces the quantization loss.

D. Voxel Fusion with 3D Attention

This module extracts the image features projected by non-empty voxels. Then, we combine the voxel sparsity information with the 3d attention mechanism to take full advantage of multi-modal features. Voxel fusion effectively reduces the dependence on the high-resolution point clouds, as described in MVX-Net. For voxel operations, we propose a 3d attention mechanism and apply the sparsity information, which are conducive to extracting effective voxelized features and emphasizing image features when voxels are sparse.

The module is composed of 3 steps. (1) The extraction of voxelized image features. (2) The 3d attention mechanism is applied to obtain the attention vectors of multi-modal voxelized features respectively. (3) We calculate the voxel sparsity, and concatenate it with image features.

In detail, we first obtain all non-empty voxels’ 8 vertex coordinates, and utilize the calibration matrix to project these point cloud coordinates to pixel coordinates in the image. Then, the largest rectangle obtained after the projection is utilized as the region of interest (RoI). Considering the different sizes of the RoIs, we use RoI Pooling to obtain 128-dimensional feature vectors from multi-scale image features.

From the discretization features obtained above, we design a 3d attention mechanism to obtain the weights of different voxels, which is shown in Fig. 3. Inspired by the spatial attention mechanism in CBAM, we respectively calculate the average and maximum features of all voxels and perform the concatenation operation. Then, combined with the voxel coordinates, the 3d sparse convolutions are performed to generate an N-dimensional attention vector, where N represents the number of non-empty voxels. The same 3d attention mechanism is applied to the voxel features obtained by VFE.

In addition, we calculate the sparse value inside the voxel to optimize the image attention vector:

$$W_i = F(MLP \left[ l_1^{d1}, \ldots, l_1^{d128}, \text{sigmoid} \left( \frac{1}{N_{points}} \right) \right])$$

(2)

$N_{points}$ represents the number of point clouds in a voxel, $l_1^d$ indicates the image feature, $W_i$ represents the attention weight for image data, $F$ denotes the operation of 3d attention and $MLP$ denotes the multilayer perceptron.

The main advantage of the proposed 3d attention mechanism is the selection of the effective voxelized features, thereby adaptively balancing the multimodal feature weights and emphasizing the image features when voxels are sparse. What’s more, the fusion on voxels reduces the dependence on the lidar point clouds.

E. SECOND Network

The SECOND [17] network improves VoxelNet and refines 3d convolution into 3d sparse convolution. First, we use the sparse conv layer and FPN to process the fused voxelized features. The structure of submanifold convolution is applied in this layer to limit the sparsity of the output, thereby greatly reducing the calculation of the convolution operation. Next, the region proposal network generates 3d proposals from the outputs of sparse conv layers. Then, after regression and refinement, the 3d detection results are generated.

IV. EXPERIMENT RESULTS

A. Implementation Details

Network Settings: The image feature extraction takes images with the resolution of 1280 × 384 as inputs. We apply ResNet-50 to subsample the image features and output the feature maps of four blocks, of which the dimensions are 256, 512, 1024, 2048. Then, FPN is applied as the image neck network, which outputs five sets of 256-d multi-scale features. For lidar point clouds, the ranges are [0, 70.4], [-40, 40] and [-3, 1] meters respectively along the X, Y and Z axis, while the voxel size is [0.05, 0.05, 0.1]. The raw features of point clouds are xyz coordinates and reflectivity. The Dynamic VFE extracts 64-dimensional voxelized features from raw features. The anchor sizes of pedestrians, cyclists and cars are respectively [0.6, 0.8, 1.73], [0.6, 1.76, 1.73] and [1.6, 3.9, 1.56] meters.

Training Details: Adam with decoupled weight decay is adopted to optimize the network. The learning rate and weight decay are set as 0.003 and 0.01. The momentum factors are 0.95 and 0.99. What’s more, we utilize warm-up for the first 1000 steps with the initial learning rate $1e^{-5}$. The total epoch is 36 and the batch size is 2. All experiments are based on the open source 3d detection toolbox mmdetection3d [18] with GPU NVIDIA GeForce GTX 1080Ti.
B. Results on KITTI Dataset

We evaluate our method on the KITTI Object Detection Benchmark [19]. This dataset contains both 2d and 3d annotations of cars, pedestrians and cyclists. There are 7481 training samples and 7518 testing samples. Following the common division rule in [7], the training samples are divided into 3712 samples as the training set and 3769 samples as the validation set. The evaluation is on the validation set for all three object categories.

We evaluate 3d object detection performance in accordance with the official KITTI evaluation protocol. For cars, 70% overlap of the 3d bounding box is required, while for pedestrians and cyclists, 50% overlap is required. Depending on different sizes, occlusions and truncations, the evaluation has three levels, that is easy, moderate and hard. The average precision (AP) at different levels are respectively calculated for the comparison.

Table I shows the performance of our method on the KITTI validation set, compared with other state-of-the-art methods. Considering that most methods only report the performance on the car category, we perform the comparison on the car category. Compared with the baseline MVX-Net, improvements in 3d and BEV are 6.9% and 6.4% respectively in hard mode. Compared with the 2d proposal method F-PointNet [3] and the 3d proposal method MV3D [7], the performance of our proposed framework has improvement in all three modes, especially in the hard mode.

In our analysis, the 2d proposal approaches focus on image features, which leads to weak processing capabilities in complex situations with more occlusions. The 3d proposal methods are overly dependent on point cloud data, resulting in the poor detection for the objects with less point cloud sampling. However, in our method, applying multiple attention fusion of point-wise and voxel-wise methods can reduce the mechanisms can reasonably select image data and point cloud situations with more occlusions or less point cloud sampling, which also enables our method to achieve better performance in hard mode.

Detection results are shown in Fig. 4. According to the comparison of column 2, 3, and 4, our method can better detect objects with low point cloud sampling, such as pedestrians in the distance. From the comparison in the first column, our method reduces false detections, which are caused by similar point cloud shapes.

C. Ablation Study

Ablation experiments are conducted to evaluate the effects of the 2d attention and 3d attention modules. All ablation studies are conducted on the pedestrian and cyclist classes, considering that these modules have a great improvement for objects with less point cloud sampling, as demonstrated before.

We report the comparison results in Table II. We first incorporate the 2d attention mechanism on the point fusion module, which increases pedestrian detection by 3% and cyclist detection by about 4%, in easy mode. In addition, there are also improvements in moderate and hard modes. This shows the effectiveness of the 2d attention mechanism.

We observe that combining the point fusion and voxel fusion modules, the detection results have not been greatly improved, which demonstrates that simply combining the above two modules cannot effectively optimize the detection performance. However, the integration of voxel fusion and 3d attention mechanism performs notably better both in pedestrian and cyclist detections, manifesting the importance of 3d attention mechanism for the voxel fusion.

Then, we investigate the effect of fusing the above four modules, that is respectively applying the 2d attention and 3d attention to the point fusion and voxel fusion module. Table II shows that this approach gets the best result. Compared with the baseline experiment, the pedestrian detection is improved by 5% and cyclist detection is improved by 4%. This shows that 2d attention and 3d attention mechanisms are beneficial for the fusion of image features, voxel features and point cloud features.

<table>
<thead>
<tr>
<th>Method</th>
<th>AP_{3D} (Car)</th>
<th>AP_{BEV} (Car)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV3D L&amp;I [7]</td>
<td>71.2</td>
<td>56.6</td>
</tr>
<tr>
<td>MV3D L [7]</td>
<td>71.3</td>
<td>62.7</td>
</tr>
<tr>
<td>F-PointNet L&amp;I</td>
<td>83.8</td>
<td>70.9</td>
</tr>
<tr>
<td>VoxelNet L</td>
<td>82.0</td>
<td>65.5</td>
</tr>
<tr>
<td>MVX-Net L&amp;I</td>
<td>85.5</td>
<td>73.3</td>
</tr>
<tr>
<td>Our Proposed Method L&amp;I</td>
<td>86.4</td>
<td>76.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Point Fusion</th>
<th>2d attention</th>
<th>Voxel Fusion</th>
<th>3d attention</th>
<th>AP_{3D} (Pedestrian)</th>
<th>AP_{3D} (Cyclist)</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>64.2624</td>
<td>58.2686 52.6567</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>61.2489</td>
<td>56.6993 51.9426</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>62.7588</td>
<td>57.7813 51.9383</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>60.2497</td>
<td>54.6029 50.3430</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>59.2110</td>
<td>55.5011 50.6946</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>57.5124</td>
<td>52.5746 49.4193</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>52.8792</td>
<td>51.2457 48.5160</td>
</tr>
</tbody>
</table>

479
V. CONCLUSION

A multi-fusion framework is proposed in this paper, which implements the attentive fusion on image features and lidar data. We propose the 3d attention mechanism for point cloud data to amplify the effective voxelized features and contributes to emphasizing the image features when voxels are sparse. This framework retains detailed image features without overly relying on the effectiveness of lidar point clouds. Experiments show that the framework can better detect the distant or small objects and effectively reduce false detections caused by similar point cloud shapes.

REFERENCES


SADA: Improved Data Symbolization and Optimization Method on HAR from Microscopic Perspective

1st Huichao Men  
School of Computer Science and Engineering  
Northeastern University  
Shenyang, P.R.China  
candyji@sina.com

2nd Botao Wang  
School of Computer Science and Engineering  
Northeastern University  
Shenyang, P.R.China  
wangbotao@cse.neu.edu.cn

Abstract—Nowadays, human activity recognition (HAR) becomes a hot topic with broad applications. Some researches have conducted HAR from microscopic perspective and achieved good results. In this article, two methods are proposed for further improvement. Firstly, an improved symbolization method with stacked sparse autoencoder is proposed for better data symbolization. Secondly, an improved multi-classification Adaboost is proposed to further optimize the recognition effect, and it is more suitable for the application scenario of this article. In the experiments section, firstly, experiments and analysis about various influencing parameters are conducted, then comparison experiments with several new or representative methods are carried out, and finally five representative sensor activity datasets (UCI Sports and Daily dataset, Wisdm Phoneacc&Watchacc dataset, Skoda dataset, HAPT dataset) are used to prove the universal applicability and achieve satisfactory effect.

Index Terms—Human Activity Recognition, Data Symbolization, Ensemble Learning, Machine Learning

I. INTRODUCTION

Human activity recognition is one of the hottest research topics nowadays. In this article, we mainly conduct research on sensor data. In some previous researches [1] [2], human activity recognition combining with NLP models from microscopic perspective has been proved meaningful in the recognition of variable length activities, and it has achieved good effect. Figure 1 is an example of decomposing one activity from microscopic perspective. In order to further improve the effect, we conduct deeper research on data symbolization about these sub-segments, and Adaboost is improved for better recognition effect, too.

Fig. 1. One Macroscopic Activity Decomposed to a Microscopic Sequence

In the previous research, the main problem is that the data symbols in an activity sequence are messy and the regularity is weak. Data symbol plays a core role in the process of migrating from NLP to HAR. The messy symbols are equivalent to noise points, which is negative on activity sequence recognition. The logical and regular symbols have an positive effect on recognition. Therefore, some researches made some efforts on feature extraction and different symbols to improve regularity. However, it still has limited effect on improving sequence recognition.

Figure 2 is a significant comparison between messy and regular activity symbols in a activity sequence.

Figure 2a shows that a messy symbol connection line like noise polyline, almost without regularity. However, the regular symbol connection line (Figure 2b) is more smooth, it is easier to recognize. The horizontal axis represents the serial numbers of points in a sequence and the vertical axis represents the symbol of one certain point. For example, point 25 to 35 in Figure 1a can be symbolized as [e,e,b,e,b,f,e,d,d,a,a], while the same part in Figure 2b is [d,d,b,b,b,b,b,b,b,b]. It is obvious that the symbol sequence in Figure 1b is more regular, concise, and easier to recognize.

The symbol points are mainly converted from the data sub-segments through feature extraction, so a good feature extraction method [3] beneficial to recognition is very important. Sparse autoencoder(SAE) is a kind of neural network that is usually used in feature extraction. Since the activity data collected by sensor is noisy, and the sparse autoencoder is more suitable for extracting features from the noisy data, a feature extraction method based on stacked SAE with good performance is used to extract features from raw data. Adaboost is a powerful approach to improve classification results with weak classifiers [4]. Therefore, Adaboost is improved in this article to optimize the classification results and improve the regularity of symbols.

Fig. 2. Comparison Between Messy and Regular Symbols
effect is proposed in this article.

Since the obtained data symbols are regular and logical, like natural language, many NLP recognition models could be migrated to HAR. In order to improve the effect of basic recognition models, Adaboost [4], an ensemble learning model is used in this article. It is usually used to optimize other machine learning models. In this article, it is further improved to adapt to the application scenario studied in this article.

In this paper, these contributions are proposed:
(1) Propose a new unsupervised human activity data symbolization method to improve the symbol regularity and logic, including stacked SAE with L-BFGS and clustering symbolization algorithm. In-depth theoretical analysis of the choice of clustering algorithm is conducted, too.
(2) An improved Adaboost with multi-classification is proposed. Two weight-setting choices are discussed.
(3) These two parts above form a complete model. Compared with the previous research, the new model has a better recognition effect.
(4) Comparison experiments of different parameters and verification experiments on multiple datasets are conducted, proving the effectiveness and universal applicability of the proposed structure. LSTM is one of the most common models applied to sequence classification and recognition. Therefore, it is used as a tool to verify the two parts of method proposed in this article.

II. RELATED WORK

Data symbolization is one of the research focuses in this article. Feature extraction is the main part of it. In many articles, statistical features [5] [6], time and frequency features [7] [8], deep learning feature extraction method (such as convolution feature [9]), etc., are the popular feature extraction methods. They all do not perform best in the research of this article. Covariance matrices [10], sliding window, enlarging window, PCA, etc., are methods aiming to expand the amount of information. Sparse autoencoder is a feature extraction model based on neural network and it is suitable for noise data. In this article, it is improved to extract better features.

Adaboost, a sequence ensemble learning method, is usually used as a tool to optimize various machine learning models. In this article it is used to optimize activity recognition models. Many classic multi-classification Adaboost models [4] convert a binary-classification into a multi-classification, such as Adaboost.MH, Adaboost.M1, and Adaboost.M2. They all have disadvantages. SAMME is a multi-classification Adaboost with unsatisfactory stability [11]. GAdaBoost is a state-and-art method with higher effectiveness but lower accuracy [12].

III. PRIOR KNOWLEDGE

In some previous researches [2], activities are decomposed into sub-segments and converted to data symbols with relevance and logic. This logical relationship is similar to that in natural language. Therefore, models from NLP could be migrated to human activity recognition.

Activity-unit is the smallest unit in the activity recognition. Original activity data is decomposed to sub-segments, then converted to activity-unit with feature extraction and symbolization. It is similar to "alphabet". Activity-combination represents a group of related combined activity-units with semantic meaning and it reflects the relevance and logical relationship between sub-segments. It is similar to "word". Activity-sequence is a sentence composed of activity-combinations, which is similar to "sentence".

Activities are converted to these forms of data similar to natural language. The logical relationship of natural languages is migrated to human activity recognition field. In our research, we will conduct further exploration based on it.

IV. METHODOLOGY

Figure 3a is the overview of the whole model in this article, named SADA. It means that SAE and Adaboost are the most important components in this model. The upper part is the overall process of it. The two big boxes are detailed figures of the data symbolization method and the improved Adaboost method. Details will be introduced in this section.

Firstly, the original training/test data is converted to data symbols(activity-units) through the improved data symbolization method. These activity-units compose regular and logical activity-sentences. Then, training activity-sentences are trained with the model optimized by improved Adaboost. Finally, the test activity-sentences are recognized by the trained model.

A. Human Activity Data Symbolization Method Based on Unsupervised Multi-layer Stacked SAE

As shown in Figure 3b, the structure in this section consists of two parts: stacked SAE(combining with L-BFGS) and clustering symbolization method. Because the sub-segments in activities are all unlabeled data, unsupervised feature extraction algorithm is necessary to generate data symbols(activity-units).

Feature extraction is the core part of data symbolization in this section. Because it is necessary to use unsupervised algorithm to generate data symbols, the output layer (such as softmax layer) in common application of SAE is removed. Data symbols are converted from feature data with clustering/classification algorithms.

In practical applications, the feature extraction effect of single-layer SAE is unsatisfactory. This article proposes a deep learning method of multiple layers stacked SAE to extract more proper features of activity data sub-segments layer by layer. As shown in Figure 3b, multiple SAEs are cascaded in multiple layers. The feature data $h_{i,j}$ of the previous layer is transferred to the next layer $h_{i+1,j}$ to continue to extract features. Finally the most representative features are obtained and the dimension is reduced. L-BFGS is used as the parameter optimization algorithm in this method. Through this method, the original data is converted to more representative feature data. The theoretical analysis of numbers of layers is described in Section 4..
Then, the feature data is converted to activity-unit (data symbol) with clustering algorithm. The principle is to generate diversified activity-units and minimize the amount of scattered points (noisy points). Classification symbolization algorithm usually corresponds to the clustering algorithm which has been chosen. Theoretical analysis is described in Section 4.3.

The activity-units generated in this part compose activity-sequences. They are very concise, regular and logical, which are suitable for human activity recognition.

B. An Improved Adaboost for Optimizing the Human Recognition Model

After the symbolization operation, the regular and concise activity-units converted from the activity data will be applied to the recognition models migrated from NLP. In this section, Adaboost is used to optimize these basic recognition models. Because of its decision function, basic Adaboost is usually used for binary-classification. It is not suitable for multi-classification. Therefore, improving the decision function is feasible to convert the basic Adaboost to a multi-classification Adaboost.

Figure 3c shows the process of the improved Adaboost in this section.

Here is the decision function of basic Adaboost. It is a binary classifier.

\[ \text{Final Classifier}(x) = \text{sign} \left( \sum_{n=1}^{N} A_n \text{Weak Classifier}_n(x) \right) \]

\( A \) is the weak classifier weights generated from iterative training.

In this section, multiple same activity recognition models with dynamic sample weights are set as the weak classifiers. Then, they are integrated into a stronger classifier. In experimental exploration, it is found that although the average recognition accuracies of some weak classifiers are declined, the recognition of different categories has achieved complementary effects, so the overall recognition accuracy can be improved.

The method in this section includes two parts. Primary Adaboost weight and fine-tuning weight could both obtain good accuracy, and the fine-tuning one is better adapt to the actual application scenarios of this article. Primary weight is a weight array generated from iterative training without fine-tuning.

In the training period, the weak classifiers are trained in the form of iteration and sequence concatenation. Weak classifiers and primary weight array, \( A \), are obtained in this period.

1) Improved Adaboost with Primary Weight: In decision function of test period, a group of weak classifiers’ results based on test data have been obtained. Then it is converted to an one-hot matrix (Formula (3)).

This group of formulas is the basic decision function in this article:

\[ A = [a_0, a_1, ..., a_{n-1}] \] \hspace{1cm} (2)

\[ \text{Sub Prediction Onehot} = [0, ..., 1], ..., [0, ..., 1] \] \hspace{1cm} (3)

\[ \text{Result Matrix} = A \times \text{Sub Prediction Onehot} \] \hspace{1cm} (4)

\[ \text{Result Matrix Sum} = \text{sum} (\text{Result Matrix}[\text{row}]) \] \hspace{1cm} (5)

\[ \text{Final Label} = \text{max index} (\text{Result Matrix Sum}) \] \hspace{1cm} (6)

\( A \) is the Adaboost primary weight array of weak classifiers. \( \text{Sub Prediction Onehot} \) is the one-hot matrix of weak classifiers’ recognition results. It is a \( n \times m \) matrix. \( n \) is the number of weak classifiers, setting as 4 here. \( m \) is the number of activity categories. \( \text{Result Matrix} \) is the cross product of \( A \) and \( \text{Sub Prediction Onehot} \). \( \text{Result Matrix Sum} \) is an array which is the result of \( \text{Result Matrix added row by row} \). \( \text{Final Label} \) is the maximum index of it.

2) Improved Adaboost with Fine-tuning Weight: Adaboost with primary weight has achieved good results. By analyzing the experiment data, in order to further improve the recognition effect, a fitting function is applied in this part.

Through the analysis of the parameters generated by multiple weak classifiers based on three datasets (UCI Dataset, Wisdm Phoneacc/Watchacc Dataset), an average empirical constant scale array of the primary weight can be estimated. Since this constant scale array is just estimated and only shows the trend, it does not represent the most appropriate weight ratio for each dataset. Therefore, we adjust the trend of estimated ratio and primary weight to fine-tune the weak classifiers weight array.
The image of the relationship between the transformed primary weight array and average constant scale array looks like the trend of tangent function (Figure 4). The final fine-tuning weight is adjusted by approximate function to make it close to the estimated ratio and suitable for the actual situation of each dataset. We try to use the transformation of the tangent function as the form of fine-tuning function.

According to the characteristic of the primary weight in basic Adaboost, it more depends on the distribution form of the data. Since the activity types collected by most common sensor activity datasets are similar, the data distribution patterns are also similar. Therefore, the trend of average scale array in many activity datasets are relatively stable. Therefore, the fitting method here is not just an empirical method, it can be applied in many datasets and has obtained acceptable results.

Tangent function is just one choice of fitting function. Maybe there are other functions suitable, too.

In fact, the average constant scale is just a tool for analysis and it is not applied in actual calculation in this model.

In Figure 4, three polylines represent the weight trend of three datasets, which is just shown as a similarity expression of the trend. The original weight line has been transformed into the form in this figure by stretching and rotating without changing the trend of the turning point. The only one curve is the standard trend of tangent function. The vertical line represents the average scale array. The horizontal line represents the weight axis after stretched and rotated operation. The center cross lines are to assist in marking the coordinate axis of the tangent function. It does not indicate the real position of the coordinate axis of three polylines and is only a formal representation. Therefore, this figure shows that the trends of the primary weight of the three datasets all conform to the tangent trend.

\[
AN = \text{normalize\_sort}(A) \tag{7}
\]

\[
k = \text{degrees}(\arctan(\frac{AN_0 - AN_{n-1}}{2})) \tag{8}
\]

\[
AW_i = \tan(\frac{\pi \times k}{\max} + AN_i) + \frac{AN_0 + AN_{n-1}}{2} \tag{9}
\]

Formula (7)(8)(9) compose a fitting function group to fine-tune primary weight. \(\max\) and \(\min\) are the maximum and minimum value of the primary weight array. \(AW\) is the fine-tuning weight array. \(n\) is set as 4 here, too. Degree function converts radians to angles. Based on the fine-tuning weight, the \(A\) in Formula 1 can be replaced by \(AW\).

Here are the total steps of the improved Adaboost in this section:

1. The whole recognition model is trained with the concise and regular training activity-sequences based on dynamic sample weights. Primary weight array \(A\) and weak classifiers are obtained after iterative training.
2. If primary weight is chosen for the improved Adaboost in this section, go to Step(4), otherwise go to Step(3).
3. \(A\) is converted to \(AW\) through the fitting function group.
4. One test activity-sequence converted by the improved data symbolization is tested in weak classifiers and the recognition results group is converted to an one-hot matrix. The \(\text{Result\_Matrix}\) is the cross product of weight array and one-hot matrix.
5. The final label is the index of the maximum of the array which is the result of \(\text{Result\_Matrix}\) added row by row.

After this process, the final label of the activity is obtained. The time complexity of the improved Adaboost is in the same order of magnitude as the basic Adaboost.

C. Analysis about Selection of Clustering and Classification Symbolization Method

Clustering algorithm is an important part of data symbolization. In this article, the selection principle is to ensure the diversity of activity-units while minimizing the number of scattered points. Figure 5 shows the difference between different algorithms.

Center-based and density-based clustering algorithms are the most commonly clustering algorithms. Meanshift and KMeans are two representative center-based algorithms. As shown as Figure 5b, because Meanshift does not set the clustering number, it clusters most points into one or few clusters, which makes it difficult to separate different activities and to distinguish between activities because of rare kinds of points(activity-units), although it looks concise. And KMeans most meets the selection principle as shown as Figure 5a. High parameter-tuning requirement makes it easy for density-based algorithms to produce scattered points labeled as -1, as Figure 5c. DPC, a state-of-the-art density-based algorithm, faces the same problem in Figure 5d.

Finally, KNN is chosen for classification because of its corresponding algorithm, KMeans.

D. Analysis about Sub-segment Length for Data Symbolization & Number of Stacked SAE Layers

Sub-segment Length for Data Symbolization: It is an important parameter in data symbolization. For NLP sequence recognition, a longer sequence contains more logical information between activity-units and it is beneficial to recognition.
In order to obtain a longer sequence, the length of sub-segment must be shorter for a constant length of data.

**Number of Stacked SAE Layers:** The original SAE is just a single-layer structure, which cannot meet the requirement of this research. The multi-layer structure is more conducive to feature extraction, but after reaching a certain value, the increasing layer number have little effect on feature extraction, and the computational cost will increase.

### V. Experiments

Experiments in this section are taken on 5 sensor datasets, including UCI Sports and Daily Dataset(19 categories) [13], WISDM Phoneacc&Watchacc Dataset(18 categories) [6], HAPT Dataset(6 categories) [14] and Skoda Dataset(10 categories) [15]. UCI dataset is applied in Section 5.1&5.2 &5.3. Skoda dataset is applied in Section 5.3.

In this section, WISDM Dataset is a composition structure of "combination of main categories and sub-categories". Distinction between main categories and the similarity between sub-categories is very obvious. It is more suitable for two-layer recognition model, so a set of single body sample is just for demonstration here.

"QC Numbers" in Table 1&2 represents "the number of categories with accuracy over 0.8".

In this section, LSTM is chosen as the migrated NLP recognition model to verify the proposed method. According to the previous research, LSTM is a proper tool to recognize concise and regular sequence with balanced efficiency and accuracy. Other NLP model may be proper, too, but it is not the focus in this article.

#### A. Comparison Experiments of Different Sub-segment Length & Different Numbers of Stacked SAE Layer

This part of the experiments in Table 1 verifies the parameter analysis in Section 4.4. From the upper part of Table 1, it can be known that a small and proper length value is more suitable for the HAR in this research. From the lower part of Table 1, it shows that enough and proper layers are beneficial to feature extraction and activity recognition. 4-layers-structure achieve the best, but 3-layers-structure is also satisfactory. A 4-layer structure is used in later experiments.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Average Acc</th>
<th>QC Numbers</th>
<th>TOP1 Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>WL</td>
<td>0.708</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>PL</td>
<td>0.382</td>
<td>12</td>
<td>1</td>
</tr>
<tr>
<td>CL</td>
<td>0.784</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>DL</td>
<td>0.802</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

#### B. Progressive Ablation Experiments and Comparison Experiments

The progressive ablation experiments include two parts. Firstly, since the core part of data symbolization is feature extraction. Two latest feature extraction methods (WISDM Feature [6] and PSD [8]) and classical convolve feature [9] are compared with the proposed data symbolization in this article (in Table 2). The recognition tool is simple LSTM.

Secondly, Adaboost.M2 [11], a classical multi-classification model, is compared with simple LSTM and the improved Adaboost proposed in this article.

In Table 2, (1)WL=WISDM Feature+LSTM (2)PL=PSD Feature+LSTM (3)CL=Convolve Feature+LSTM (4)DL=Proposed Data Symbolization in this article+LSTM (5)DML=Proposed Data Symbolization in this article with Adaboost.M2+LSTM (6)SADA=Proposed Whole Model in this article. According to the conclusions in article [1], these methods are suitable for comparison.

In this section, the ablation experiments prove the effectiveness and advantages of the two parts of the proposed structure. WISDM Feature and PSD Feature both do not archive satisfactory effects. The method based on convolution has been greatly improved, but it is still poorer than the proposed data symbolization method. Simple recognition model(LSTM) without optimization and Adaboost.M2 are also less effective than the proposed optimization method in this article.

#### C. Additional Comparison Experiments

The experiments in last section have verified that the two parts of the method proposed in this article has a better effect based on previous research.

ILVote [16] is a latest human activity recognition model based on incremental learning and vote. Transformer [17] is one of the hottest models nowadays in various research fields, to prove the proposed method in this article. BERT [18] is a latest model developed from Transformer, mainly used for NLP. In this section, they are taken as comparisons without feature extraction to prove the advantage of our whole proposed method.

The main shortcoming of ILVote is highly dependent on manual parameter setting, especially the full activity window length. In many cases, it is difficult to determine a proper full activity window length. Therefore, although the theoretical basis of ILVote is meaningful, it is lack of universal applicability.
Because of difficulty to set a proper window length, the results are unsatisfactory.

In the absence of proper feature extraction in this part of experiments, the training and test sequences imported into the Transformer and BERT are very irregular and lack of logic. Even this state-and-the-art models can not recognize them well. Therefore, it verifies that the concise and regular sequence generated by the proposed method in this article is meaningful.

As shown as the experiments results in Table 3, after obtaining regular and logical sequence, LSTM, a most common NLP model, could achieve good recognition effects.

D. Verification Experiments with Primary Weight and Fine-tuning Weight

This section is a group of verification experiments based on the whole structure proposed in this article with primary weights and fine-tuning weights on 5 datasets. In Table 5(next page): (1) UCI=UCI Sports and Daily Dataset (2) WPA=WISDM Phoneacc Dataset (3) WWA=WISDM Watchacc Dataset (4) SK=Skoda Dataset (5) HA=HAPT Dataset (6) FT=Fine-Tuning (7) AA=Average Accuracy

From the Table 4, the proposed model has achieved good results with primary weights. In case of model with fine-tuning function on some datasets, although the accuracy of a small numbers of categories decrease, more categories’ accuracy increase, and the average accuracy increase, too. For example, in the experiments results of Skoda and HAPT, only one activity’s accuracy decreases, and the average accuracy and the accuracies of most activities all increase.

Experiments in this section prove the universal applicability and effectiveness of the proposed structure both with primary weights and fine-tuning weights.

VI. CONCLUSION AND FUTURE WORK

Analyzing activity from microscopic perspective with migrated NLP methods is the foundation of our research. In this article, we propose a new data symbolization method and an improved Adaboost for basic recognition models with good effect and universal applicability. Both of them need be further improved. In the future, we will continue to take more research on this topic and try to further improve the recognition effect.

VII. ACKNOWLEDGEMENT

This work is supported by “the Fundamental Research Funds for the Central Universities” (N2104002).

REFERENCES


HARP Pro: Hierarchical Representation Learning based on global and local features for social networks

Wei Zhang, Jing Yang¹, Fanshu Shang
College of Computer Science and Technology
Harbin Engineering University
Harbin, China

Abstract—The purpose of network representation learning methods is to learn the node low-dimensional representations to accomplish node classification, link prediction, network visualization, and so on. Most of the network representation learning methods cannot keep local structural features and global structural features, resulting in poor performance at classification tasks for social networks. To solve this problem, we propose HARP Pro, a novel method for learning latent node representations which can maintain lower-order and higher-order hierarchical structures. On the one hand, HARP Pro coarsens the graph based on the community detection method. It can preserve the relationships between bridge nodes and communities. On the other hand, it presents a graph coarsening method based on Degree to keep the relationships between nodes and their neighborhood. It can capture the lower-order hierarchical structure of the graph. Then it puts the hierarchical information into the network embedding methods iteratively like HARP. Finally, it obtains node representation vectors which integrate global and local structural features. Experimental results on CiteSeer and Blogcatalog dataset show that the performance of HARP Pro is better than HARP and the baseline methods, DeepWalk, LINE, and Node2vec. The results reveal that HARP Pro can sustain local and global structural features.

Keywords—social network; network embedding; community; graph coarsening; hierarchical structure

I. INTRODUCTION

We are in the era of Big Data. Because of Big Data, there are countless ties between everyone and everything. So long as has the place which the ties exist, the network exists, such as social network, protein network, logistics network, communication network, and electricity network. Getting appropriate representations for nodes and exploring the network is important for content recommendation[12], information diffusion[13], disease-related genes prediction[14], and resource assignment[15]. Hence, network embedding has received considerable attention recently.

The social network is a kind of large-scale network which has massive, sparse data with noise. The complex network data is a major challenge to the traditional network analysis methods. Inspired by word2vec[9], DeepWalk[1] captures path information with random walks. It regards the paths as sentences and puts them into the model to learn the representations. It gets rid of the relational matrix. Thus, its time complexity is low. However, with the limitation of the walk length, DeepWalk can only get the local context information of nodes. Thus, representations it learns just reflect the local structure of the graph. Unlike the depth-first sequence of DeepWalk, Node2vec[3] combines depth-first strategy with breadth-first strategy, enriches the sample space. This method is still lumbered with the walk length. Therefore, the path information it gets is still limited in the neighborhood of nodes. LINE[2] tries to mine the node relationships with first order similarity and second order similarity to deal with the problem. But this method just expands the sample space to the second order area. The global structure cannot be captured by LINE.

Based on the above facts, HARP[4] learns the embedding in smaller graphs which are obtained by coarsening the original graph. In this way, the global structure can be embedded into the original graph iteratively. HARP thinks that the shape of the input graph is the global structure. Then it maintains this shape in the process of graph coarsening. Unfortunately, the external shape is unable to fully reveal the internal hierarchical structure.

Community structure is ubiquitous in social networks. It is a known higher-order hierarchical structure. The relationships between communities constitute the entire network. And inside the community, hub nodes and bridge nodes string together and support the communication in the community. Inspired by this knowledge, we propose HARP Pro, a general meta-method to maintain the lower-order and higher-order hierarchical structures of social networks for network embedding.

First, we proposed a local structure coarsening method based on Degree[10] to maintain the lower-order hierarchical structure. Then, we presented a global structure coarsening method based on a community detection method to keep the higher-order hierarchical structure. Based on these two methods, we can get a set of coarse-grained graphs. Finally, we embed the node representation of the higher-order graph into the lower-order graph to get the final network representation with global and local features.

II. RELATED WORK

HARP[4] method consists of three parts: graph coarsening, graph embedding, and graph representation refinement. The process of HARP is listed in Table I.

¹Corresponding author: Jing Yang (email: yangjing@hrbeu.edu.cn).
DOI reference number: 10.18293/SEKE2021-145.
In Step 1, HARP provides two graphs coarsening strategies. One is edge collapsing. They introduce the edge collapsing algorithm[5] to coarsen the equivalent edges. This strategy can keep the peer-to-peer structure. The other is the star collapsing. This strategy merges the leaf nodes to sustain the shape of the star topology.

<table>
<thead>
<tr>
<th>TABLE I. HARP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
</tr>
<tr>
<td>The original network ( G_0 = (V_0, E_0) )</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
</tr>
<tr>
<td>node representations of ( G_0, \phi )</td>
</tr>
</tbody>
</table>

| Step 1. Coarsen \( G_0 \) to get smaller graphs \( G_1, G_2, \ldots, G_L \). |
| Step 2. Put \( G_2 \) in EMBED() to get the node representations of \( G_2, \phi_{G_2} \). |
| Step 3. For \( i=L-1 \) to 0 do |
| Step 4. Set \( \phi_{G_i} \) as the input embedding of network \( G_i \) and learn the node representations \( \phi_{G_i} \). |
| Step 5. end for |
| Step 6. return \( \phi_{G_L} \) |

### III. HARP Pro

HARP Pro consists of three parts: graph coarsening, graph embedding, and graph representation refinement. It is the same with HARP. The difference between HARP Pro and HARP is the process graph coarsening. In order to capture the lower-order and higher-order hierarchical structures, we develop two kinds of graph coarsening methods.

As we have mentioned above, the community is the higher-order hierarchical structure of social networks. From a macro perspective, communities and the relationships between them form a giant net, which can make information spread through the whole network. Bridge nodes play a vital role in transmission between communities. From a micro perspective, the hub nodes and their followers form the community. Then the neighborhood of the community member is the lower-order hierarchical structure. From these two kinds of hierarchical structures, we need to coarsen the network based on hub nodes and bridge nodes. Hub node owns a large number of followers. We can find them with Degree. Hence, we present a local structure coarsening method based on Degree.

#### A. A local structure coarsening method based on Degree

The major steps of the graph coarsening method for local structure are listed in Table II.

First, we lock the hub node with Degree. Then, we start the process of coarsening with these nodes. By merging the hub node and its neighbors, we can coarsen the first-order neighborhood structure of the original graph \( G_0 \) and get a new coarse-grained graph \( G_1 \). Running this algorithm on graph \( G_1 \), we can coarsen the second-order neighborhood structure of the original graph \( G_0 \). The output of the algorithm is the new coarse-grained graph \( G_2 \).

#### B. A global structure coarsening method based on community structure

Utilizing the above method, we can sustain the local structure in the coarse-grained graphs. For the global structure, we introduce Louvian[6], a community detection method to get the community structure of the original graph. Louvian believes that there are more links inside the community. And outside the community, links are more sparse.

Firstly, the community structure is employed to find bridge nodes. Then, we keep them and set them as the new nodes in the new coarse-grain graph. The rest nodes of the community will be merged, and regard as the new node in the new graph. In this way, we can obtain the higher-order hierarchical structure of the original graph. This process is listed in Table III.

<table>
<thead>
<tr>
<th>TABLE II. COARSENING METHOD FOR LOCAL STRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The network ( G_i )</td>
</tr>
<tr>
<td><strong>Output:</strong> This coarse-grained graph, ( G_i )</td>
</tr>
</tbody>
</table>

| Step 1. Sort the nodes in \( G_0 \) with Degree in descending order, and put the result in the list, DL; |
| Step 2. For each node \( v \) in DL do |
| Step 3. Compare the degree of node \( v, dv \) with the max degree of its neighbors, \( dv_m \); If \( dv > dv_m \) do |
| Step 4. Combine node \( v \) with the rest of its neighbors, set them as a new node in \( G_i \), and remove these node from DL; |
| Step 5. If \( dv= dv_m \) do |
| Step 6. The neighbor nodes with the max degree will be kept in in \( G_i \). Combine node \( v \) with the rest of its neighbors, set them as a new node in \( G_i \), and remove these nodes from DL; |
| Step 7. return \( G_i \) |

<table>
<thead>
<tr>
<th>TABLE III. COARSENING METHOD FOR GLOBAL STRUCTURE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> The original network ( G_0 )</td>
</tr>
<tr>
<td><strong>Output:</strong> This coarse-grained graph, ( G_i )</td>
</tr>
</tbody>
</table>

| Step 1. Use Louvian to get the community attribute of each node; |
| Step 2. For each community \( com \) do |
| Step 3. Find the bridge nodes in community \( com \), set them as new nodes in \( G_i \); |
| Step 4. Merge the rest nodes of community \( com \), set them as a new node in \( G_i \); |
| Step 5. return \( G_i \) |

### C. The Framework of HARP Pro

Based on Table II and Table III, we can get three coarse-grained graphs, \( G_1 \), \( G_2 \), and \( G_3 \). After this procedure, we can start the graph embedding and graph representation refinement. The framework of HARP Pro is listed in Table IV.

In Step 1, we run the two graph coarsening method on the original network \( G_0 \) respectively, then we can get \( G_1 \) with the first-order structure, \( G_2 \) with second-order structure, and \( G_3 \) with community structure. In Step 3, we start with \( G_1 \) to learn its node representations based on the embedding method. In Step 4-7, we prolong and refine the representations. If a node in \( G_1 \) also appears in \( G_2 \), we replace this node representation in \( G_2 \) with \( G_1 \)'s. Otherwise, node \( a \) in \( G_2 \) is merged into node \( b \) in \( G_3 \), then \( a \) should have identical weights in word2vec as \( b \). Then the new representation of \( G_2 \) is obtained. We go through the same process on \( G_1 \) and \( G_0 \). Finally, we can get new node representations of \( G_0 \).
TABLE IV. HARP PRO

| Input:                  | The original network, $G_0$  
|                        | Arbitrary graph embedding algorithm, EMBED(). |
| Output:                | The node representations of $G_0$, $\varphi$ |
| Step 1: $G_0, G_1, G_2, G_3 \leftarrow \text{GRAPHCOARSENING}(G_0)$; |
| Step 2: Initialize $\varphi_{G_3}$ by assigning zeros; |
| Step 3: $\varphi_{G_3} \leftarrow \text{EMBED}(G_3, \varphi_{G_3})$; |
| Step 4: For $i = 2$ to 0 do |
| Step 5: $\varphi_{G_i} \leftarrow \text{PROLONGATE}(\varphi_{G_{i+1}}, G_{i+1}, G_i)$; |
| Step 6: $\varphi_{G_i} \leftarrow \text{EMBED}(G_i, \varphi_{G_i})$; |
| Step 7: end for |
| Step 8: return the node representations of $G_0$, $\varphi$ |

IV. RESULTS AND DISCUSSIONS

A. Data Preparation

Two real-world networks are used to evaluate the performance of different methods. They are CiteSeer[7] and BlogCatalog[8].

CiteSeer is a kind of citation network. In this network, nodes represent the authors of papers. Edges are reference relationships between papers. The labels indicate the research area of the paper. Papers in this network are classified into six kinds: AI, IR, ML, DB, Agents, and HCI.

BlogCatalog is a social network of the BlogCatalog website. Bloggers are the nodes. Edges represent the social relationships between users. The labels indicate the interests of bloggers. The interests are divided into 39 classes.

The information about these two datasets is shown in Table V.

TABLE V. STATISTICS OF NETWORK DATASETS

<table>
<thead>
<tr>
<th>Network</th>
<th>$N$</th>
<th>$E$</th>
<th>$C$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CiteSeer</td>
<td>3312</td>
<td>4732</td>
<td>6</td>
<td>Classification</td>
</tr>
<tr>
<td>BlogCatalog</td>
<td>10312</td>
<td>333983</td>
<td>39</td>
<td>Classification</td>
</tr>
</tbody>
</table>

$N$, the number of nodes; $E$, the number of edges; $C$, the number of categories; $T$, multi-label classification task.

B. Baseline Methods

We use the following network embedding methods to conduct three parallel experiments:

DeepWalk —— DeepWalk uses random walk to capture the path information of nodes. It treats route sequences as sentences and puts them into the Skip-gram model to learn the node embeddings.

LINE —— LINE maps nodes to the vector space based on the density of node relationships. It combines the first-order relationship with the second-order structure to project nodes with a strong connection to a similar location based on the objective function. The Skip-gram model is applied to solve the objective function.

Node2vec —— To improve DeepWalk, Node2vec change the way of random walk. By tuning the return parameter $p$ and in-out parameter $q$, it can explore the node neighborhood with DFS and BFS. The Skip-gram model is also used in this method.

C. Parameter Settings

For each baseline method, we embed it with HARP and HARP Pro to compare the classification performance. Hence, we conduct three parallel experiments: (1) DeepWalk, HARP(DeepWalk), and HARP Pro(DeepWalk); (2) LINE, HARP(LINE) and HARP Pro(LINE); (3) Node2vec, HARP(Node2vec) and HARP Pro(Node2vec).

- Deepwalk Group: The number of random walks is set as 40. The walk length $t$ is 10 and the window size is 5. We set the representation size $d$ as 128.
- LINE Group: The representation size $d$ is set as 64, and the iteration time is 50.
- Node2vec Group: The number of random walks, walk length, window size and representation size are the same as the DeepWalk group. We set the in-out parameter and the return parameter as 1.0.

D. Graph Coarsening

No matter how large the network is, HARP Pro just needs three times coarsening. One time is to get the global structure. The second time is to capture the local feature of the original graph. But for HARP, the coarsening time depends on the scale of the network. For CiteSeer dataset, HARP needs 18 times to reach the preset size. And for BlogCatalog, 24 times coarsening is needed to remain the shape of the network unchanged. We used Gephi, a network topology visualization tool to draw the graphs that we got by HARP Pro.

![Figure 1. The graph coarsening effect of HARP Pro on CiteSeer dataset.](image-url)
Fig. 1 and Fig. 2 show the graph coarsening effect of HARP Pro on the two datasets. $G_0$ is the original graph. $G_1$ and $G_2$ are the subgraphs that we got based on Table II, and $G_3$ is the subgraph with the community structure that we got based on Table III.

As shown in Fig. 1, CiteSeer network is formed with multiple ring structures. With two times graph coarsening based on Algorithm 2. The longitudinal ring structures are kept in $G_1$ and $G_2$. The ring structures in the other orientation are in decline. In $G_3$, a few ring structures remain. From $G_0$ to $G_3$, the scale of the network drastically reduced.

In Fig. 2, BlogCatalog dataset demonstrates a star topology. With the reduction in the number of nodes, this feature becomes more obvious.

In general, the graph coarsening method in HARP Pro can maintain the shape of the original graph.

E. Multi-label Classification

Classification is the most common application of the network embedding method. In order to observe the performance of the classification task, we choose some nodes and their labels randomly as the training dataset. The proportion of the training dataset is varied from 10% to 90% in this experiment. To predict the labels of the remaining nodes, we train a one-vs-rest logistic regression model with L2 regularization. The logistic regression model is implemented by LibLinear[11].

Fig. 3 and Fig. 4 report Marco F1 scores of the three parallel experiments.

**CiteSeer.** For the group of DeepWalk, as the size of labeled data increases, the Marco F1 score of three methods ascends step by step basically. By contrast, our method owns the highest score all the time. When the ratio of labeled nodes is small(10%-50%), the scores of the three methods are close. But our method is still higher than HARP and Deepwalk. The relative gain of our method is over 1%. When the ratio of labeled data is big(60%-90%), the advantages of our method are obvious. The relative gain of our method is over 5% with 90% labeled data. For the group of LINE, the trend of the curve is similar to the DeepWalk group. The difference is that the score of our method and HARP is relatively close. But our method also consistently outperforms with the relative gain 1%.
For the group of Node2vec, our advantage is more obvious. When the ratio of labeled data is small (10%-60%), the relative gain of our method remains close to 2% through that period. When the ratio of labeled data becomes bigger, HARP shows a downward trend. However, the rising rate of our method is even greater. In general, our method maintains the leading position in three parallel experiments.

**BlogCatalog.** For the group of DeepWalk, both three methods show the tendency to ascend. The score of the three methods is close. But the relative gain of our method is 1% with 10% and 80% labeled data. As for the LINE group, the situation is a bit different. When the ratio of labeled data is small (10%-40%), the scores are close. But when the ratio of labeled data becomes bigger, the difference between our method and HARP becomes wider. The relative gain of our method is over 3% with 70% labeled data. HARP and our method reveal the trend of a fast increase in Node2vec group. Our method also performs better than Node2vec and HARP. Generally, the accuracy of HARP Pro in classification is better performance than HARP and the other baseline methods.

**F. Discussion**

Compared with Fig. 3 and Fig. 4, we can see the advantage of our method in CiteSeer is more obvious than BlogCatalog for the multi-label classification tasks. As shown in Fig. 2, Blogcatalog data owns plenty of star structures. Star structure is not a typical community structure. Because the links inside and outside it are both sparse, the star structure cannot be distinguished by the community detection method. The global structure cannot be kept in the graph completely. As for CiteSeer, the distribution of links between nodes is relatively uniform. Community structures can be well identified with Louvain. Thus, the advantage of HARP Pro on BlogCatalog is smaller.

In summary, HARP Pro is good at learning node representation of the network with significant community structures.

**V. Conclusion**

From a microcosmic perspective, nodes and their neighbors form their own communities. From a macro perspective, the entire network is constituted by a number of communities. Inspired by this fact, we propose a networking embedding method based on community structure and neighborhood structure, HARP Pro. HARP Pro starts with Degree to coarsen the first-order and second-order neighborhood structure. It maintains the local hierarchical information in the subgraph. Then, HARP Pro merges nodes with community features. It captures the high-order hierarchical structure. Finally, it obtains node representation vectors by capturing global and local structural features. Experimental results on CiteSeer and Blogcatalog dataset show that HARP Pro performs better than HARP and the other baseline methods.

Because of the instability of the community detection method, we introduce more uncertain factors to the model. In the future, we would like to find a more stable approach to capture the community feature. And we will take into account the node influence and the role of the node to learn the network representation.

**Acknowledgment**

This paper is supported by the National Natural Science Foundation of China (No. 61672179, 61370083 and 61402126), the Natural Science Foundation of Heilongjiang (No. F2015030), the Distinguished Young Scholars of Heilongjiang (No. QC2016083), the Postdoctoral Science Foundation of Heilongjiang (No. LBJ-Z14071).

**References**


A Volume-Aware Positional Attention-Based Recurrent Neural Network for Stock Index Prediction

Xinpeng Yu
School of Electronics and Computer Engineering
Peking University
Shenzhen, China
yuxinpeng@pku.edu.cn

Dagang Li*
International Institute of Next Generation Internet
Macau University of Science and Technology
Macau, China
dagang.li@ieee.org

Abstract—With the rapid development of deep learning, more researchers have attempted to apply nonlinear learning methods such as recurrent neural networks (RNNs) and attention mechanisms to capture the complex patterns hidden in stock market trends. Most existing approaches to this task employ an attention mechanism that primarily relies on the information extracted from input features but fails to consider the other important factors (e.g., trading volume and position), which can potentially enhance these attention-based approaches. Motivated by the observation, we extend the attention mechanism with features needed for stock performance prediction in this article. Specifically, we propose a volume-aware positional attention-based recurrent neural network (VPA-RNN) for this task. First, we propose a generic method of adding position awareness to the attention mechanism. Next, the trading volume is incorporated into the original attention distribution to form a revised distribution. To evaluate the effectiveness of VPA-RNN, we collected real stock market data for stock indexes S&P 500 and DJIA, and the experimental results show that the proposed VPA-RNN can significantly outperform several existing highly competitive methods.

Keywords—stock index prediction; recurrent neural network; attention mechanism; volume-aware attention; positional attention

I. INTRODUCTION

Stock performance prediction has received much attention due to its decisive role in stock investment, which aims to predict the future price or trend of stocks in order to achieve the maximum profit from stock investment. Various methods have been proposed to predict stock performance by many economic analysts and stock traders.

Li et al. [1] applied a quantile AR model to analyze the dynamics of stock index returns in China. In addition, the hidden Markov model (HMM) has been used to make nonlinear predictions of stock trends. Zhang et al. [2] presented an approach to predict stock market price trends based on a high-order HMM for the purpose of considering both short and long-term time dependence. However, such traditional solutions have apparent drawbacks, as they lack the capability of modeling the nonstationary and nonlinear nature of stock prices. To address this issue, many methods based on deep learning have been proposed to forecast stock prices in recent years. More researchers have attempted to apply deep learning methods such as multilayer perceptions (MLPs) [3] and recurrent neural networks (RNNs) [4–6] to capture the complex patterns hidden in market trends. Although the traditional RNN is capable of processing nonlinear data, it is not sufficient to model long-term dependence on a time series. This motivates the use of gated memory cells; thus, the famous long short-term memory (LSTM) network was proposed to better model long-term dependency on a time series and mitigate the vanishing gradient problem [7]. Accordingly, many studies employ the LSTM neural network in financial prediction [8–10].

However, if the time series is very long, LSTM also suffers from the problem of vanishing gradients which results in decreasing performance [11]. To overcome this problem, researchers have proposed the attention mechanism that achieved great success in various fields, including neural machine translation [12], speech recognition [13], and image processing [14]. Therefore, several recent works introduced an attention mechanism to stock-related applications [15–22]. Li et al. [16] proposed a multi-input LSTM (MI-LSTM) model, which can extract valuable information from low correlation factors and discard their harmful noise by employing additional input gates controlled by the convincing factors called mainstream. Furthermore, Qin et al. [17] proposed the dual-stage attention-based RNN (DA-RNN), drawing inspiration from the encoder-decoder structure used in machine translation. The DA-RNN model predicts the stock index of the next day using the previous values of stock indexes and constituent stock prices as input. This model consists of an encoder and a decoder. The encoder is composed of LSTM and an input attention mechanism that is used to adaptively extract the relevant features at each time step by referring to the previous encoder hidden state. The output of the encoder serves as the input of the decoder. The decoder is composed of LSTM and a temporal attention mechanism that is used to select the relevant encoder hidden states across all time steps. In this way, the DA-RNN model can not only adaptively select the most relevant input features but also capture the long-term temporal dependencies of a time series appropriately.

In the abovementioned attention-based stock price prediction model, the temporal attention mechanism primarily relies on the information extracted from input features but fails to consider the other important factors (e.g., trading volume and position), which can potentially enhance these attention-based approaches. Motivated by the observation, we extend the temporal attention mechanism...
mechanism with features needed for stock performance prediction in this article. On the one hand, the position of the time step is a key factor in the stock performance prediction task. It is natural that the time steps closer to the predicted time step are more important. However, the abovementioned attention methods take no account of the effects of positions of different time steps, i.e., identical or very similar time steps are scored equally regardless of their positions in the sequence. Therefore, we introduce the positional attention mechanism to the task of stock performance prediction, which has achieved success in various fields, including natural language processing (NLP) [23,24] and speech recognition (SR) [25]. On the other hand, trading volume is also an important feature that provides valuable information, as past trading volume predicts both the magnitude and the persistence of future price momentum [26], i.e., the time step with higher trading volume is generally more important, and the attention mechanism should pay more attention to such time steps. Thus, inspired by several task-oriented attention mechanisms [27,28], we take advantage of this feature of stock performance prediction and propose volume-aware attention to incorporate the trading volume into the original attention distribution to achieve attention recalibration.

Combining these two gives a volume-aware positional attention-based recurrent neural network (VPA-RNN) with markedly better stock index prediction performance. To justify the effectiveness of the VPA-RNN, we compare it with the state-of-the-art approach using the S&P 500 dataset and the DJIA dataset. Our proposed VPA-RNN achieves the RMSE that is 6.80% and 47.83% lower than that of the best previous model DA-RNN [17], respectively.

II. RELATED THEORY AND TECHNOLOGY

In this section, we introduce the LSTM and the attention mechanism, which are the foundations of both the proposed model and the comparative models in this article.

A. Long short-term memory neural networks (LSTM)

Due to its memory blocks, the LSTM network [7] has a strong capability of capturing the long-term memory of sequential data with high prediction capability on chaotic time series. Hence, many related works adopt LSTM to learn long-term temporal dependencies from stock data time series [16,17]. For a similar reason, we also use LSTM in this paper. LSTM is a variant of RNNs that uses a gating mechanism to control the flow of information into or out of memory. For convenience, in this study, we use the function LSTM(·, ·) as shorthand for the LSTM model in (1):

\[ (h_t, c_t) = \text{LSTM}(x_t, h_{t-1}, c_{t-1}, W, b), \]  

where \( W \) and \( b \) include all of the weight matrices and bias vectors, which are determined in the training process.

B. Attention mechanism

Based on recurrent neural networks, sequence-to-sequence models (S2S) have become popular due to their success in machine translation [29]-[31], which is composed of encoder and decoder. The encoder is used to convert the input sentences into a fixed-length vector and then used by the decoder to produce output sequences. However, encoder-decoder networks encounter the long-term dependency problem that their performance will deteriorate rapidly as the length of the input sequence increases. To resolve this issue, the attention mechanism is employed to select parts of hidden states across all the time steps by allocating adequate attention to key information [12]. The general attention mechanism is often implemented by scoring each encoder hidden states \( h_i \) in \( H = (h_1, h_2, \ldots, h_n) \) separately based on the previous decoder hidden state \( s_{t-1} \) and normalizing the scores \( e_{ij} \) by a softmax function to generate the attention weight \( \alpha_{ij} \):

\[ e_{ij} = \text{Score}(s_{t-1}, h_i), \]  
\[ \alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{i=1}^{n} \exp(e_{ij})}. \]

Then, the decoder input \( c_i \) at \( i \) is the weighted sum of \( h_j \) and calculated as follows:

\[ c_i = \sum_{j=1}^{n} \alpha_{ij} h_j. \]

Under the attention mechanism, the dependencies between the source and target sequences are not restricted by the intermediate distance. Consequently, it is helpful for overcoming the long-term dependency problem, and it was soon extended into various fields, including stock-related applications [15]-[22].

III. VOLUME-AWARE POSITIONAL ATTENTION-BASED RNN

In this section, we first introduce the notation used in this article and the problem we aim to study. Then, the motivation and details of the proposed VPA-RNN model for stock index prediction are presented.

A. Notation and Problem Statement

The goal of this work is to predict the closing price of the next day. Given the previous values of the target as \( Y = (y_1, y_2, \ldots, y_T)^\top \in \mathbb{R}^T \) where \( T \) represents the size of the time window and \( y_t \) is the target at time \( t \). Similarly, the time series of all features would be denoted as \( X = (X_1, X_2, \ldots, X_T)^\top \in \mathbb{R}^{T \times N} \) where \( N \) specifies the number of features. Hence, \( X_t = (x_{t1}, x_{t2}, \ldots, x_{tN}) \in \mathbb{R}^N \) is a vector of all the \( N \) features at time \( t \) and \( X^n = (x^n_{t1}, x^n_{t2}, \ldots, x^n_{tN}) \in \mathbb{R}^N \) is the time series of the \( n \)-th feature in time window \( T \). Thus the VPA-RNN model aims to learn a nonlinear mapping function \( F(\cdot) \) as follows:

\[ \hat{y}_{T+1} = F(y_1, y_2, \ldots, y_T, X_1, X_2, \ldots, X_T). \]

The features used in this paper include open, close, high, low, adj_close, and volume in the granularity of the trading day. Adj_close is an abbreviation of the adjusted closing price, which amends a stock’s closing price to accurately reflect that stock’s value after adjustments for splits and dividend distributions. Deemed as the true price of stocks, it is often used when examining historical returns or performing a detailed analysis of historical returns. Therefore, this study uses adj_close of the next day as the target \( Y \).

Among all the features, only the feature volume does not belong to the type of stock price, which refers to the number of
transactions in a trading day. Specifically, we represent the historical series of \textit{volume} as \( V = (v_1, v_2, ..., v_T) \in \mathbb{R}^T \), and it is used to achieve attention recalibration in the next subsection.

### B. Proposed Model

The overall structure of our proposed VPA-RNN model is shown in Fig. 1. Inspired by existing work, we employ a dual-stage attention-based encoder-decoder neural network. In the encoder, we introduce an input attention mechanism proposed by [17], which is used to select the relevant features adaptively. In the decoder, our proposed volume-aware positional attention is used to automatically select relevant encoder hidden states across all time steps. With the help of the proposed attention mechanism, the decoder can take account of the effects of volumes and positions of different time steps in order to assign weight more appropriately.

\[ \alpha^n_t = \frac{\exp(e^n_t)}{\sum_{i=1}^N \exp(e^n_i)}, \]  

where \( \alpha^n_t \) is the attention weight measuring the importance of the \( n \)-th input feature. Finally, we can adaptively select the features as follows:

\[ \tilde{X}_t = (\alpha^n_1 x^n_1, \alpha^n_2 x^n_2, ..., \alpha^n_T x^n_T)'. \]  

Thus, (6) can be updated as:

\[ h_t = \text{LSTM}_1(h_{t-1}, \tilde{X}_t), \]

where \( X_t \) is replaced by \( \tilde{X}_t \) that considers the weights of different features. Therefore, the encoder can adaptively select certain features rather than pay attention to all the features equally.

#### 2) Decoder with volume-aware positional attention

In the decoder, we use another LSTM to decode the encoded input information. In order to adaptively select relevant encoder hidden states across all time steps, we employ a temporal attention mechanism. Specifically, the attention weight \( l^i_t \) of each encoder hidden state is calculated based on the previous decoder hidden state \( d_{t-1} \in \mathbb{R}^m \) and cell state \( c_{t-1} \in \mathbb{R}^p \):

\[ l^i_t = v^\top_d \tanh(W_d[d_{t-1}; c_{t-1}] + U_d h_t), 1 \leq i \leq T, \]

where \([d_{t-1}; c_{t-1}] \in \mathbb{R}^{2p}\) is a concatenation of the hidden state and cell state of the previous LSTM unit. \( v_d \in \mathbb{R}^m, W_d \in \mathbb{R}^{m \times 2p}, \) and \( U_d \in \mathbb{R}^{m \times p} \) are parameters to be learned. We omit the bias terms here for clarity. The attention weight \( \beta^i_t \) that represents the importance of the \( i \)-th encoder hidden state \( h_t \) is calculated by the formula:

\[ \beta^i_t = \frac{\exp(l^i_t)}{\sum_{j=1}^T \exp(l^j_t)}. \]  

However, such temporal attention mechanism suffers from two problems: (1) identical or very similar time steps are scored equally regardless of their positions in the sequence. But the position of each time step is a key factor in the task of stock performance prediction. (2) It does not explicitly model the effect of volume of each time step in the input sequences, which is another important feature that provides valuable information. Therefore, we propose a new volume-aware positional attention mechanism to tackle these challenges, as shown in Fig. 2, which can evaluate the relative contribution of each time step not only on the information of encoder hidden states but also on the global position and volume of each time step.

First, inspired by the position encoding vectors used in [23], we define a position vector \( p_i \) for each time step in the time window \( T \) as follows:

\[ p_i = \left( \frac{i}{T}, \frac{i}{T}, ..., \frac{i}{T} \right) \in \mathbb{R}^m, 1 \leq i \leq T, \]

where \( m \) is the dimension of position encoding vectors, that is the same as the size of the decoder hidden state in order to facilitate calculation, then we add the position encoding vector to the calculation formula of \( l^i_t \), and (11) is updated as follows:

\[ l^i_t = v^\top_d \tanh(W_d[d_{t-1}; c_{t-1}] + U_d h_t + E_d p_i), 1 \leq i \leq T, \]
Figure 2. Our proposed volume-aware positional attention mechanism.

where $E_d \in \mathbb{R}^{m \times m}$ is a learnable parameter. In this way, the position encoding vector $p_t$ can provide important spatial information of each time step.

Second, inspired by the study of trading volume in [26], we argue that the volume of a time step to some extent reflects the importance of this time step. Hence, during the learning process of the attention mechanism, the effect of volume should be considered explicitly, we change (14) by multiplying $l_i^t$ by the volume of this time step $v_t \in \mathbb{R}$, and the updated formula is as follows:

$$l_i^t = v_t^T \tanh(W_d[d_{t-1}; c_{t-1}'] + U_d h_t + E_d p_t) v_t,$$

then, when the trained model is used to test on the test sequences, $v_t$ will be replaced by the volume of the corresponding time step in the test sequences. In this way, the model can learn the attention weights under the condition that the effect of volume is considered explicitly, and the original learned attention distribution without being multiplied by the volume $v_t$ can exclude the effect of the trading volume. Then, it can be multiplied by the volumes of the corresponding time steps in the test sequences to achieve attention recalibration when testing.

The context vector $C_t$ is a weighted sum of all the encoder hidden states $h_t$, and it is distinct at each time step in the decoder.

$$C_t = \sum_{i=1}^{T} \beta_i h_i.$$  

Then, the given target series $Y = (y_1, y_2, ..., y_T) \in \mathbb{R}^T$ can be combined with the context vectors:

$$\tilde{y}_t = \tilde{w}^T [y_t; C_t] + b,$$

where $\tilde{w} \in \mathbb{R}^{m+1}$ and $b \in \mathbb{R}$ are used to map the concatenation $[y_t; C_t] \in \mathbb{R}^{m+1}$ to the size of the decoder hidden states. The newly calculated $\tilde{y}_t$ is used to update the decoder hidden state:

$$d_t = \text{LSTM}_d(d_{t-1}, \tilde{y}_t).$$

Finally, the final prediction result can be obtained by a linear function:

$$\hat{y}_{T+1} = v_y^T (W_y [d_T; C_T] + b_w) + b_v,$$  

where $[d_T; C_T] \in \mathbb{R}^{p+m}$ is a concatenation of the last decoder hidden state and context vector, and $p$ is the size of the decoder hidden states. $W_y \in \mathbb{R}^{p \times (p+m)}$ and $b_w \in \mathbb{R}^p$ are the parameters used to map the concatenation to the size of the decoder hidden states. The linear function with weights $v_y \in \mathbb{R}^p$ and bias $b_v \in \mathbb{R}$ produces the final prediction result.

### C. Training procedure

The minibatch stochastic gradient descent (MGD) and Adam optimizer are used for the model training with mean squared error (MSE) as the loss function:

$$\text{loss} = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_{T+1} - y_{T+1})^2.$$  

### IV. Experiments

In this section, we first describe the two datasets used for empirical studies. Then, we present the experimental setup, including parameter settings and evaluation metrics. Finally, we conduct extensive experiments to evaluate the performance of the proposed VPA-RNN by comparing it to different baselines.

#### A. Datasets and Setup

We collected real-world historical data of two stock indexes from the Yahoo! Finance website: the S&P 500 Index (S&P 500) and the Dow Jones Industrial Average Index (DJIA), traded from Jan 3rd, 2000, to Dec 30th, 2020, at a daily frequency, for a total of 21 years, to test the effectiveness of the proposed VPA-RNN model. For a fixed time window of size $T+1$ and a stride of 1, each sample incorporated input sequences of $T$ time steps and a target index value for model training and evaluation. All data from both datasets were normalized to between 0 and 1 using min-max scaling. Then, we divided each dataset into 7:1.5:1.5 ratios in the time dimension as the training set, validation set, and test set. The basic process of evaluation was to use the training set to train the model and obtain a classifier every epoch. Then, the best classifier was selected based on the validation set and was finally evaluated on the test set.

#### B. Parameter Settings and Evaluation Metrics

In the experiments of the previous study [17] for DA-RNN, the length of time window 10 yielded the best results. Hence, in this study, all the compared models use the same length of time window as 10 for comparison. In addition, other parameter settings for the baseline models are selected based on optimal experimental results. Other parameter settings of our VPA-RNN model are shown in Table I.

To measure the effectiveness of various methods for stock index prediction, we consider three different evaluation metrics: root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination ($R^2$). Specifically, the smaller the RMSE and MAE, the closer the predicted value to the true value; the closer the coefficient $R^2$ to 1, the better the fit of the model.
\[ MAE = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|, \quad (21) \]

\[ RMSE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2}, \quad (22) \]

\[ R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}, \quad (23) \]

where \( y_i \) is the true value, \( \bar{y} \) is the mean value of all true values, and \( \hat{y}_i \) is the predicted value.

C. Results

To evaluate the effectiveness of the proposed VPA-RNN, we conduct experiments to compare our results with those of the compared models, including a standard long short-term memory neural network (LSTM), the encoder-decoder network (Encoder-Decoder) proposed in [32], we change it to perform stock index prediction as Qin et al. did in [17], and the dual-stage attention-based recurrent neural network (DA-RNN) proposed in [17]. Furthermore, we compare our proposed VPA-RNN model against the setting that only employs its positional attention mechanism without the volume-aware attention mechanism (PA-RNN) and the setting that only employs its volume-aware attention mechanism without the positional attention mechanism (VA-RNN). All models take the same input for a fair comparison. For all the compared methods, we train them ten times and report their average performance. The comparison results of all the models over the two datasets are shown in Table II.

As illustrated in Table II, the DA-RNN model outperforms Encoder-Decoder, which has no attention layer, indicating the effectiveness of the dual-stage attention mechanism since it is capable to adaptively extract relevant features and select relevant hidden states across all time steps. In addition, our proposed PA-RNN, VA-RNN, and VPA-RNN all show better performance than Encoder-Decoder and DA-RNN. This suggests that taking the position of each time step into account and extending the attention mechanism to be volume-aware can both provide more reliable attention weights to make more accurate predictions. With the integration of the positional attention as well as the volume-aware attention, our proposed VPA-RNN achieves the best MAE, RMSE, and \( R^2 \), that increase of 11.76\%, 6.80\%, and 0.41\% and 56.81\%, 47.83\%, and 15.82\% for the S&P 500 and DJIA datasets, respectively, compared to the DA-RNN model, indicating the effectiveness of our overall model structure.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameter Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>lr</td>
<td>Learning rate</td>
<td>0.001</td>
</tr>
<tr>
<td>epoch</td>
<td>Number of epochs</td>
<td>1000</td>
</tr>
<tr>
<td>batch_size</td>
<td>Batch size</td>
<td>128</td>
</tr>
<tr>
<td>encoder_lstm_unit</td>
<td>Neuron number in encoder LSTM</td>
<td>64</td>
</tr>
<tr>
<td>decoder_lstm_unit</td>
<td>Neuron number in decoder LSTM</td>
<td>64</td>
</tr>
<tr>
<td>activation</td>
<td>Activation function</td>
<td>Tanh</td>
</tr>
</tbody>
</table>

TABLE II. STOCK INDEX PREDICTION RESULTS OVER THE S&P 500 DATASET AND DJIA DATASET (BEST PERFORMANCE DISPLAYED IN BOLDFACE)

<table>
<thead>
<tr>
<th>Models</th>
<th>S&amp;P 500 Dataset</th>
<th>DJIA Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( MAE ) ((\times 10^{-2}))</td>
<td>( RMSE ) ((\times 10^{-2}))</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.96</td>
<td>1.41</td>
</tr>
<tr>
<td>Encoder-Decoder</td>
<td>1.28</td>
<td>1.75</td>
</tr>
<tr>
<td>DA-RNN</td>
<td>1.02</td>
<td>1.47</td>
</tr>
<tr>
<td>PA-RNN</td>
<td>0.98</td>
<td>1.42</td>
</tr>
<tr>
<td>VA-RNN</td>
<td>0.94</td>
<td>1.38</td>
</tr>
<tr>
<td>VPA-RNN</td>
<td>0.90</td>
<td>1.37</td>
</tr>
</tbody>
</table>

For visual comparison, we show the prediction results of Encoder-Decoder, DA-RNN, and VPA-RNN over the DJIA dataset in Fig. 3. We can see that our proposed VPA-RNN generally fits the ground truth much better than Encoder-Decoder and DA-RNN, which shows the proposed volume-aware positional attention mechanism is indeed effective in the problem of stock index prediction.

![Graph showing stock index prediction results](image)

Figure 3. DJIA Index vs. Time. Encoder-Decoder (top) and DA-RNN (middle) are compared with VPA-RNN (bottom).

V. Conclusion

In this paper, we note two important factors (e.g., trading volume and position), which can potentially enhance the
attention mechanism for stock index prediction. Motivated by the observation, this study proposes a novel volume-aware positional attention recurrent neural network (VPA-RNN). Specifically, we add a position vector for each time step in the input sequences into the calculation formula of attention score to take the important spatial information into account. Then, we incorporate the trading volume into the original attention distribution to achieve attention recalibration. Based upon these two improvements, the VPA-RNN can take advantage of the features of stock index prediction and thus provide more reliable attention weights to make more accurate predictions. Extensive experiments on the S&P 500 dataset and the DJIA dataset demonstrated the superior performance of the proposed VPA-RNN relative to the original LSTM, Encoder-Decoder, and DARNN, indicating the VPA-RNN model has broad application prospects and is highly competitive. In summary, this work provides new insight into attention-based stock index prediction research and can help to develop better predicting models.

In the future, we will investigate whether feeding more technical indicators and basic information or adding predictions based on stock-related news can result in more accurate predictions. Furthermore, it is also promising to apply the proposed model to more granular trading data, such as hourly or per-minute transaction data.

**ACKNOWLEDGMENT**

This work was supported by National Key R&D Program of China (2019YFB1804400).

**REFERENCES**


DeepSCC: Source Code Classification Based on Fine-Tuned RoBERTa

Guang Yang Yanlin Zhou Chi Yu Xiang Chen∗
School of Information Science and Technology, Nantong University, Nantong, China
1930320014@stmail.ntu.edu.cn 1159615215@qq.com yc_struggle@163.com xchencs@ntu.edu.cn

Abstract

In software engineering-related tasks (such as programming language tag prediction based on code snippets from Stack Overflow), the programming language classification for code snippets is a common task. In this study, we propose a novel method DeepSCC, which uses a fine-tuned RoBERTa model to classify the programming language type of the source code. In our empirical study, we choose a corpus collected from Stack Overflow, which contains 224,445 pairs of code snippets and corresponding language types. After comparing nine state-of-the-art baselines from the fields of source code classification and neural text classification in terms of four performance measures (i.e., Accuracy, Precision, Recall, and F1), we show the competitiveness of our proposed method DeepSCC.

1 Introduction

Recently, multiple programming languages (such as Java, Python, C++) are often used together in the large-scale software development process, since different development tasks often use different programming languages. When developers ask questions on Stack Overflow [1] [2], the answers to the questions are closely related to the type of programming language. Therefore, Stack Overflow needs to use the correct programming language tag of posts to match the related answers for users, and the source code classification task can effectively solve this problem.

In the previous studies, this task is often modeled as a text classification problem. Then machine learning methods can be used to classify the source code’s language type. For example, Khasnabish et al. [3] used a Naive Bayesian classifier. Alrashedy et al. [4] used a random forest classifier and XGBoost. Motivated by the research progress of neural text classification and code semantic learning [5], we propose a novel method DeepSCC by fine-tuning the pre-trained model RoBERTa [6] to perform the source code classification task.

To verify the effectiveness of our proposed method DeepSCC, we choose a corpus collected from Stack Overflow, which contains 224,445 pairs of code snippets and corresponding language types. We first perform data processing on this corpus, such as word segmentation, discarding noisy code snippets. Then, we use the corpus to fine-tuning the RoBERTa model [6]. We compared DeepSCC with nine state-of-the-art baselines. For these chosen baselines, two baselines are selected from the source code classification field [4, 7], and the remaining baselines are selected from the neural text classification field [8–12]. In terms of four performance measures (Accuracy, Precision, Recall, and F1), we find DeepSCC can outperform these baselines.

The main contributions of our study can be summarized as follows:

(1) We propose a novel method DeepSCC by fine-tuning the pre-trained model RoBERTa [6], which can classify the language type of the source code. We share our trained classification model for other researchers to follow and replicate our study1.

(2) We choose corpus gathered from Stack Overflow as our experimental subject. Then we choose two baselines proposed by Alrashedy et al. [4] (i.e., in the source code classification field) and seven baselines based on TextCNN and Transformer (i.e., in the neural text classification field). Final experimental results show that DeepSCC can improve the performance of source code classification.

2 Related Work

In previous studies on source code classification, Kennedy et al. [13] proposed a software language model to recognize the entire source code file from Github. Their classifier is based on five natural language statistical models. They gathered corpus from GitHub and considered 19 programming languages. Khasnabish et al. [3] collected more than 20,000 source code files. These source codes are downloaded from multiple repositories in GitHub. The model uses the Bayesian classifier and aims to predict ten programming languages. Klein et al. [14] collected 41,000 source code files from GitHub as the training set and randomly selected 25 source code files as the test set. However, their methods, which are based on supervised learning and feature selection methods, can only achieve 48% accuracy at most. Alrashedy et al. [7] proposed the method SCC to classify source code snippets via Naive Bayes classifier, with an accuracy of about 75%. This method can also distinguish programming language families (such as

* Xiang Chen is the corresponding author.
DOI reference number: 10.18293/SEKE2021-005
1https://huggingface.co/NTUYG/DeepSCC-RoBERTa
C, C# and C++) with an accuracy of 80%, and can identify programming language versions (such as C#3.0, C#4.0, and C#5.0) with an accuracy of 61%. Recently, Alrashedy et al. [4] classified the language types for code snippets in Stack Overflow. They used the random forest classifier and XGBoost to build classifiers. Different from the previous studies, we are the first to introduce a pre-trained model to this task and then proposed a novel method DeepSCC. The final results show the competitiveness of our proposed method when compared to state-of-the-art baselines.

3 Method

3.1 Overview of DeepSCC

In this section, we show the framework of DeepSCC in Figure 1. In particular, we first preprocess the corpus, including data cleaning, filtering, and word segmentation. Then we fine-tune the pre-trained model RoBERTa to predict the type of programming language.

3.2 Data Preprocessing Phase

In this phase, the data cleaning and filtering are consistent with previous work [4]. However, we find that the previous code classification methods treat the code word as the basic unit. Its disadvantage is that it cannot effectively solve the out-of-vocabulary (OOV) problem. That means there exist some words, which are not in the training set but in the testing set. To solve the OOV problem, we use the Byte-Pair Encoding (BPE) proposed by Sennrich et al. [15]. It is a mixture between character-level and word-level representations. Using BPE can avoid a large number of "[UNK]" symbols in the test set, as "[UNK]" symbols may decrease the performance of the pre-trained model. For example, the original code snippet is "def split_lines(s): return s.split(\n)", and the result after using BPE segmentation is "def", "\"split\"", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\", "\”,...
Figure 1: The framework of our proposed method DeepSCC

Figure 1: The framework of our proposed method DeepSCC

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy(%)</th>
<th>Precision(%)</th>
<th>Recall(%)</th>
<th>F1(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Forest</td>
<td>78.728</td>
<td>79.362</td>
<td>78.825</td>
<td>78.874</td>
</tr>
<tr>
<td>XGBoost</td>
<td>78.803</td>
<td>79.925</td>
<td>78.891</td>
<td>79.217</td>
</tr>
<tr>
<td>TextCNN</td>
<td>82.662</td>
<td>83.561</td>
<td>82.706</td>
<td>82.964</td>
</tr>
<tr>
<td>TextCNN+FastText</td>
<td>84.201</td>
<td>84.719</td>
<td>84.285</td>
<td>84.369</td>
</tr>
<tr>
<td>TextCNN+Word2Vec</td>
<td>84.600</td>
<td>85.071</td>
<td>84.677</td>
<td>84.764</td>
</tr>
<tr>
<td>Transformer</td>
<td>79.035</td>
<td>79.801</td>
<td>79.107</td>
<td>79.272</td>
</tr>
<tr>
<td>Transformer+FastText</td>
<td>75.624</td>
<td>76.026</td>
<td>75.526</td>
<td>75.986</td>
</tr>
<tr>
<td>Transformer+Word2Vec</td>
<td>74.325</td>
<td>75.050</td>
<td>73.243</td>
<td>73.765</td>
</tr>
<tr>
<td>BERT</td>
<td>86.946</td>
<td>87.292</td>
<td>87.004</td>
<td>87.116</td>
</tr>
<tr>
<td>DeepSCC</td>
<td>87.202</td>
<td>87.424</td>
<td>87.276</td>
<td>87.135</td>
</tr>
</tbody>
</table>

Table 1: The comparison results between DeepSCC and baselines

4.3 Baselines

In the RQ, we first compare our proposed method DeepSCC with two state-of-the-art methods from source code classification (i.e., Random Forest and XGBoost methods used in SCC++ [4]). We also choose TextCNN [8] and Transformer [11] with/without pre-trained word vectors (i.e., FastText [9] or Word2Vec [10]) from the neural text classification field as baselines. Besides, we also select BERT [12] as a baseline for the pre-trained model.

4.4 Implementation Details

In our study, we use Pytorch 1.6.0 to implement our proposed method. For baselines in the source code classification field, we run their shared code on our preprocessed corpus. For baselines in the neural text classification field, we re-implemented these baselines according to the corresponding description by Pytorch. For BERT and RoBERTa, we pre-train the model in the transformer library.

It needs to be noticed that pre-trained models (i.e., BERT and RoBERTa) use the method BPE for code segmentation by default. For other baselines, we choose the word_tokenize method provided by the NLTK library for code segmentation.

We run all the experiments on a computer with an Inter(R) Core(TM) i7-9750H 4210 CPU and a GeForce GTX3090 GPU with 24 GB memory. The running OS platform is Windows 10.

According to the analysis of the experimental results, we can find: (1) From Table 1, we can find that our method can outperform baselines and achieves the best performance in source code classification. Specifically, it can achieve a maximum performance improvement of 17%, 16%, 19%, and 18% in terms of Accuracy, Precision, Recall, and F1 respectively. The results show that the two-way transformer encoder can learn the deep semantics of the code snippets more effectively, which is helpful to obtain a better classification performance. (2) Not all the baselines in the neural text classification field outperform the baselines in the code classification field. That means some traditional machine learning methods can outperform deep learning-based methods in this task. (3) For baselines in the field of neural text classification, Transformer is not as effective as TextCNN in code classification. This may be because Transformer learns too little code semantics. After adding pre-trained word vectors (such as Word2Vec and FastText), the performance of TextCNN can be slightly improved, but the performance of Transformer is decreased. This shows that pre-trained word vectors can better capture the feature
representation of the code when the structure is CNN in this task. (4) From Table 2, we can find that DeepSCC can achieve high performance in most of the programming languages. Then we analyze the cause of the poor performance when the programming languages are C/C++ and CSS/JavaScript. Specifically, 8% of the code snippets with the actual category of C are predicted to be C++, and 10% of the code snippets with the actual category of C++ are predicted to be C. Since C++ is almost a superset of C, this indicates that some C++ code snippets and C code snippets are indistinguishable, which poses a challenge for the source code classification problem. 6% of the code fragments with the actual category of CSS are predicted to be JavaScript, and 7% of the code snippets with the actual category of JavaScript are predicted to be CSS. Because CSS as a style language often appears in the scripting language JavaScript, it is used to dynamically update page elements. This leads to the simultaneous appearance of JavaScript and CSS in the code snippets, which also poses another challenge for the source code classification problem.

5 Conclusion

In this paper, we propose a novel method DeepSCC for source code classification, which is based on fine-tuned RoBERTa [6]. To show the effectiveness of DeepSCC, we used four widely used performance measures to evaluate the performance of DeepSCC. The results show the competitiveness of DeepSCC when compared to nine state-of-the-art baselines from the fields of source code classification and neural text classification.

Acknowledgement

Guang Yang and Yanlin Zhou have contributed equally to this work and they are co-first authors. This work is supported in part by Natural science research project in Universities of Jiangsu Province (18KJB520041).

References

Multi-Granularity Code Smell Detection using Deep Learning Method based on Abstract Syntax Tree

Weiwei Xu, Xiaofang Zhang
School of Computer Science and Technology
Soochow University
Suzhou, China
Email: xfzhang@suda.edu.cn

Abstract—Code smell refers to poor design that is perceived to have a negative impact on readability and maintainability during software evolution, and it implies the possibility of refactoring. Therefore, the effective detection of code smell is of great importance. Many approaches including metric-based, heuristic-based, and machine learning approaches have been proposed to detect code smells. However, all these methods use manually selected features, which is highly subjective and difficult to select the most appropriate features. Recently, deep learning methods without extensive feature engineering have been proposed. Nevertheless, these token-based approaches may not achieve good results because they ignore many semantic and structural information of source code. To this end, we propose a novel deep learning approach based on abstract syntax trees (ASTs) to detect multi-granularity code smells, which captures the semantic and structural features of code fragments from the ASTs. The experimental results on four types of smells show that this approach achieves better results than the state-of-the-art approaches for detecting code smells with different granularities.

Index Terms—code smell, abstract syntax tree, deep learning

I. INTRODUCTION

Code smell refers to some bad designs in the code, which often has a bad effect on the readability and maintainability of the software. Furthermore, code smell suggests the possibility of refactoring [1], so detecting code smells in a timely and effective manner can be a guide for developers in refactoring. Software engineering researchers have done a lot of research on the definition, causes, and effects of code smell [2].

A number of approaches have been proposed to detect different types of code smells in source code. Metric-based [3] and heuristic-based [4] approaches are the traditional ways to detect code smells. However, most of them have strong limitations because they all rely on manually designed heuristics to obtain final results from manually selected features. Picking the most appropriate features and building heuristics are very difficult, and computing the corresponding metrics for the target source code is a considerable amount of work. In recent years, many scholars have proposed to use machine learning methods such as Support Vector Machine, Naive Bayes and Logistic Regression to detect code smells. Although machine learning methods avoid manually designed heuristics [5], existing machine learning methods for detecting code smells are still in need of further research and improvement [6]. Machine learning methods require a collection of features extracted from the source code, i.e., they still require external tools to compute many metrics of source code.

Recently, Sharma et al. experimented with a deep learning approach without extensive feature engineering to detect code smells and verified the feasibility of the approach on several smells [7]. Deep learning models can learn intrinsic features during training to classify samples, but existing deep learning methods have some limitations as follows.

- The deep learning models are token-based. The token-based code representation may lose the rich semantic and structural information in the source code.
- Existing methods focus only on code smells with small granularity, lack of experimentation on code smells with larger granularity.
- A universally well-performing deep learning model was not found for different code smells.

To address these problems, we propose a novel abstract syntax trees (ASTs) based code smell detection approach (AST-CSD). The approach extracts the ASTs from the code fragments and forms sequences of statement trees by splitting the complete AST into several sub-trees. First, we encode the sequences of statement trees and then extract semantic and structural features from the sequences using bi-directional GRU [8] and maximum pooling. Final vector representations of code fragments can be obtained after that. At last, the final detection result is derived through several fully-connected layers. We apply the AST-based approach to 500 high-quality Java projects from GitHub. Better results are achieved than the state-of-the-art deep learning models, not only on one type of small-grained code smells but also on three types of larger-grained code smells.

The main contributions of this paper are as follows.

- We propose a deep learning approach based on ASTs to detect code smells. To the best of our knowledge, we are the first to conduct research on code smell detection using deep learning methods based on ASTs.
- In addition to smells with small granularity, we focus on detection of code smells with larger granularity, bridging
the gap of previous work and enabling detection of multi-
granularity code smells.

- For different code smells, we conduct extensive experi-
ments to find out the parameter configuration that makes
the model perform best.

The rest of this paper is organized as follows. Section II
introduces the background; Our AST-CSD approach is intro-
duced in Section III; Section IV describes the experimental
setup and results are in Section V; The conclusion of this
paper and the future work are presented in Section VI.

II. BACKGROUND

A. Code Smell

Fowler and Beck first introduced the notion of code smell
[1] and defined it as “certain structures in the code that
suggest (or sometimes scream) for refactoring.” Code smell
affects the readability and maintainability of programs and has
an impact on the software development and evolution process.

Code smells can be divided into implementation [1], design
[9] and architecture [10] smells in the order of size according
to their granularity or scope [7]. Implementation Smells have
the smallest granularity and scope, and they usually occur on
methods. Design Smells, which are in the middle granularity,
typically occur at the class level. Architecture smells have the
greatest granularity, often involving multiple components, and
their impact is at the system level.

B. Abstract Syntax Tree

An abstract syntax tree (AST) is a tree representation of
the abstract syntactic structure of source code written in a
programming language [11]. The abstract syntax tree clearly
describes the structure of the source code. In many existing
studies, source code is parsed into abstract syntax trees to
produce code representations that capture the semantic rela-
tionships between different code elements [12], [13]. Code
representation based on abstract syntax trees is now being
used for code clone detection [11], defect prediction [14],
auto program repair [15], and other problems. In metric-based
code smell detection methods, abstract syntax trees may also
be used to compute a set of source code metrics [7], [16].

However, these methods do not take advantage of the rich
semantic and structural information in the abstract syntax trees.

C. Motivation

Existing deep learning methods for code smell detection are
token-based. The token-based code representation approaches
treat code fragments as natural language texts. Although
code fragments have some similarity with natural language
texts, code fragments should not be treated simply as natural
language texts because there is rich structural information in
code fragments [17]. For example, two statements located
closely to each other, one outside the loop body and one inside
the loop body, are semantically disjoint. But the token-based
approach does not reflect this disjoint relationship well.

Recent work has demonstrated the superiority of an AST-
based approach to code representation over a token-based
approach [17], [18]. Intuitively, the rich semantic and struc-
tural information in AST will help us in smell detection. For
example, when we detect the code smells such as complex
method, there are three adjacent loop statements in the method,
and the token-based method does not clearly show whether the
three loop statements are nested or not. By contrast in AST,
we can determine by observing whether the three statements
are at the same depth of the tree. Whether the loop statements
are nested or not obviously is critical to judge the complexity
of the method. Therefore, we believe that more semantic and
structural information in the AST-based code representation
approach is of great help in code smell detection.

III. APPROACH

This section introduces the method we use to detect code
smells. Figure 1 gives an overview of our method.

A. Data preprocessing

We first use the CodeSplit\footnote{https://github.com/tushartushar/CodeSplitJava} to split all the projects down-
loaded from Github into class-level and method-level code
fragments. Then we use Designite [19] to find out the smells
contained in the source code and generate smell reports. Based
on the smell report, we divide the code fragments that have
been split up to corresponding granularity into two categories, one containing smells and one without.

B. Decomposition of ASTs

We use Javalang\(^2\) to parse the code fragment and get the AST of it, and then we store the AST with its corresponding label. Figure 2 shows the decomposition of an AST, the left side of the figure shows the code fragment of a method, and the right side shows its complete abstract syntax tree. According to the method of Jian Zhang et al. [17], we split the each statement like Try statement into two parts which has a header and a body containing a lot of statements. The statement trees are marked with dashed lines in Figure 2, and the red node is the root node of the statement tree. By preorder traversal, we obtain a sequence of statement trees. We store the sequences of statement trees and their corresponding labels of whether they contain the smell, which are later used to train the ASTNN model.

C. Training of ASTNN

We use the ASTNN model proposed in [17] and Figure 3 shows the structure of it. The model includes three parts: encoding statement trees, representing statement sequences and classification.

1) Encoding Statement Trees: To obtain vector representations of statements, we use an RvNN-based statement encoder. There are many syntactic symbols in ASTs, and we obtain all the symbols in ASTs as a corpus by traversing ASTs in preorder. Then we use the word2vec [20] to learn unsupervised vectors of the symbols. Given a statement tree \( t \), let \( n \) denote a non-leaf node and let \( C \) denote the number of its children nodes. In the beginning, the lexical vector of node \( n \) can be obtained by:

\[
v_{n} = W_{e}^{\top} x_{n} \tag{1}\]

where \( W_{e} \in \mathbb{R}^{d \times k} \) is the pre-trained embedding parameters with the vocabulary size \( V \) and the embedding dimension of symbols \( d \), \( v_{n} \) is the embedding of symbol \( n \) and \( x_{n} \) is its one-hot representation. Then the vector representation of node \( n \) can be calculated using the following equation:

\[
h = \sigma(W_{n}^{\top} v_{n} + \sum_{i \in [1,C]} h_{i} + b_{n}) \tag{2}\]

where \( W_{n} \in \mathbb{R}^{d \times k} \) is the weight matrix and \( k \) is the encoding dimension, \( h_{i} \) is the hidden state of its each child, \( b_{n} \) is a bias term, \( \sigma \) is the activation function, for which we use identify function in the method, and \( h \) is the latest hidden state. We can recursively compute the vector representations of all nodes in the statement tree \( t \). Finally, we obtain the vector representation of the entire statement tree \( t \) by maximum pooling sampling:

\[
e_{t} = [\max(h_{11}), \max(h_{12}), \ldots, \max(h_{ik})], i = 1, \ldots N \tag{3}\]

where \( N \) is the number of nodes in \( t \).
2) Representing the Sequence of Statement Trees: In the previous procedure, we can get vector representations of all statement trees, so for each AST, we have a sequence of statement tree vectors. Using this sequence of statement tree vectors, we then use bi-directional GRU [8] to capture the naturalness of statements. Finally, we sample the most important features of these states by means of the max pooling. At this point, we obtain a vector representation of the code fragment.

3) Classification: After obtaining the vector representations of the code fragments, we feed them into a neural network consisting of several fully connected layers and classify them into two classes, one containing smells and one without.

IV. EXPERIMENTAL SETTINGS

A. Projects and datasets

We choose to use the same dataset, 500 high-quality Java projects covering a variety of functions from Github, as used in [7]. Since implementation smells and design smells occur at the method level and class level, respectively, and a class usually contains many methods, if the same number of projects are used for both types of smells in the experiment, the former will have a much larger sample size. Consequently, for the implementation smells with small granularity (i.e., method level), we select 100 projects randomly from 500 projects, while for smells with large granularity, we use all 500 projects. For samples with different granularity, we process them separately: removing duplicate samples and deleting overlong samples with the length over one standard deviation away from the mean. The goal of this procedure is to keep the training set within a reasonable range and avoid wasting memory and processing resources.

We divide all samples into three parts, 70% as the training set, 10% as the validation set, and 20% as the test set. To reduce the impact of the extreme imbalance, we balance the positive and negative samples in the training set. The number of both positive and negative samples in the training set is limited to 5000, and if there are more negative samples than positive samples in the training set, the number of negative samples is reduced to 5000, and if there are more negative samples than positive and negative samples in the training set. The number reduce the impact of the extreme imbalance, we balance the set, 10% as the validation set, and 20% as the test set.

To further explore the effectiveness of deep learning methods in detecting smells with different granularity, in addition to implementation smells, our experiments focus on the design smells with larger granularity which are more difficult to detect.

We choose Insufficient Modularization (IM, i.e., the class has not been completely decomposed), Deficient Encapsulation (DE, i.e., the declared accessibility of one or more members of the class is more permissive than actually required), Feature Envy (FE, i.e., the class has a method that uses methods and data of other classes more than using its own ones and seems more interested in a class other than the one it actually is in). We select these design smells because they are representative due to their high frequency of occurrence in the 500 projects.

What’s more, we choose Empty Catch Block (ECB, i.e., a catch block of an exception is empty), which is an implementation smell. This smell was also chosen in the experiments of Sharma et al. [7], and we use it to verify that our model also has a good performance on small-grained smells.

C. Baseline setting

In this paper, we select the following three baseline methods proposed by Sharma et al. [7] as comparative methods to estimate the performance of our proposed method:

1) CNN-1D Model: In this model, each input instance is represented by a 1D array of tokens. The model extracts features through convolution, batch normalization, and max pooling layers. Finally, the fully-connected layers are used to make predictions about whether a given instance belongs to the positive or negative class.

2) CNN-2D Model: The CNN-2D model is similar to the CNN-1D model, except that each input instance of CNN-2D model is a 2D array of tokens, which delineates the source code statement by statement.

3) RNN Model: The RNN model has the same input as CNN-1D, but unlike CNN-1D, RNN captures features using an embedding layer and a LSTM layer.

We obtain the hyper-parameters configurations for the baseline methods according to [7]. Tables II and III show the values of the hyper-parameters for the CNN and RNN models. All combinations of hyper-parameters are performed to confirm the best configuration of baseline methods.
D. Evaluation

Due to the extremely unbalanced distribution of positive and negative samples in real projects, we avoid comparing the accuracy of each model because if a model predicts all samples as negative, it will still have high accuracy. We choose precision, recall and F-measure as the evaluation metrics. They are defined as follows:

\[
\text{precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}
\]  
\[
\text{recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}
\]  
\[
F\text{-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

V. EXPERIMENTAL RESULTS

In this section, we mainly focus on answering the following research questions:

RQ1: How does the AST-based approach perform under different configurations for multi-granularity code smells?

RQ2: Is the AST-based approach better than the token-based approaches in detecting code smells with different granularities?

RQ3: Is the AST-based approach significantly superior?

A. RQ1: How does the AST-based approach perform under different configurations for multi-granularity code smells?

Table IV shows the values of the different hyper-parameters for our approach. We perform 64 combinations of hyper-parameters to get the best configuration of our approach.

Table V lists the performance of our AST-based approach AST-CSD on the optimal configuration. From the table, we can easily see that the AST-based approach does not perform equally on the four types of smells, and the combination of hyper-parameters that make the approach perform optimally for different smells also varies. Figure 4 shows the violin plot of performance of the approach under all configurations for four smells. Among the four smells, the AST-based method has good performance in detecting Deficient Encapsulation and Insufficient Modularization smell. However, this AST-based method did not perform very well in detecting Feature Envy smell. We believe this is somewhat related to the extremely unbalanced ratio of positive and negative samples for this smell, which reaches a ratio of 353:29845 in the test set. For Empty Catch Block smell, which is an implementation smell, it can be seen from Figure 3 that different hyper-parameters have a greater impact on the performance.

B. RQ2: Is the AST-based approach better than the token-based approaches in detecting code smells with different granularities?

We first perform parameter search for the baseline methods to find the combination of hyper-parameters that has the best performance. Table VI shows the hyper-parameters of the three token-based methods when they achieve the best performance. After getting the optimal configurations of each method, we repeat the training and testing of each approach on the optimal configuration for 30 times. Figure 5 shows the average performance of the AST-based approach compared to the token-based approaches on the four smells. As shown in Figure 5,
the AST-based approach achieves better results than the token-based approaches on all three smells with large granularity. In addition, the AST-based approach also obtains better results for smell Empty Catch Block with small granularity.

C. RQ3: Is the AST-based approach significantly superior?

To further analyze the performance of AST-based approach and baseline approaches, Wilcoxon signed-rank test and Cliff’s delta test are conducted. If \( p – value \) of Wilcoxon signed-rank test is less than 0.05, the two matched samples are significantly different. Cliff’s delta test can be used as a complementary analysis to Wilcoxon signed-rank test, and Cliff’s delta test can measure the effective level of difference between the two sets of observation data. Table VII shows Cliff’s delta values(\( \delta \)) and the corresponding effective levels.

We use the Win/Tie/Loss indicator to compare the performance of different methods. Specifically, if the AST-based method outperforms a baseline method with the \( p – value \) of Wilcoxon signed-rank test less than 0.05 and the Cliff’s delta value greater than or equal to 0.147, the difference between these two methods is statistical significant, in which case we consider AST-CSD to win. Conversely, if the baseline model is better than the AST-based method and the difference between them is significant, we consider AST-CSD to lose. In other cases, we consider them to be tied. What’s more, ‘+’ or ‘-’ before the effective level is to represent the positive or negative Cliff’s delta values. ‘+’ means the AST-based method is superior.

As shown in Table VIII, our AST-based method significantly outperforms other token-based methods in detecting all four smells with different granularities.

VI. THREATS TO VALIDITY

A. Internal validity

One of the main factors affecting internal validity is the experimental environment. We use the Designite tool to detect smells, which is used to generate labels for the training data, and view its results as ground truth. Since it is widely used in related work [7], [19], we think it is reliable to use the tool to detect code smells. In addition, the code is carefully reviewed and tested to ensure that the code we used to build the model was error-free.

B. External validity

External validity refers to the validity of generalization of research results. In this study, we use 500 open source Java projects. Experiments on other datasets (non-Java projects or industrial projects) will help to further validate our approach.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a new deep learning method based on ASTs to detect code smells. We exploit the rich semantic and structural information in the AST to generate the final feature representations of code fragments. Experiments on smells with different granularities show that our method is significantly better than state-of-the-art deep learning methods in terms of \( F – measure \).

As future work, smells with greater granularity, i. e., architectural smells, need to be considered. When using deep learning methods to detect smells with greater granularity, how to use the numerous components involved in architectural smells as input is a question worth investigating. What’s more, it is of great value to extend our approach to other programming languages, such as Python and C++.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China(61772263, 61772014, 61872177), Collaborative Innovation Center of Novel Software Technology and Industrialization, Undergraduate Training Program for Innovation and Entrepreneurship of Soochow University(202010285141H), and the Priority Academic Program Development of Jiangsu Higher Education Institutions.
REFERENCES


FCEP: A Fast Concolic Execution for Reaching Software Patches

Meng Fan\textsuperscript{1,2}, *Wenzhi Wang\textsuperscript{1,2}, Aimin Yu\textsuperscript{1}, Dan Meng\textsuperscript{1}
\textsuperscript{1}Institute of Information Engineering, Chinese Academy of Sciences
\textsuperscript{2}School of Cyber Security, University of Chinese Academy of Sciences
Beijing, China
\{fanmeng,wangwenzhi,yuaimin,mengdan\}@iie.ac.cn
* Corresponding Author: wangwenzhi@iie.ac.cn

Abstract—Software updates that bring new features to the users or that fix old errors can easily introduce new errors, which makes it necessary for users to repeatedly consider whether to update the software to the latest version. Therefore, the security testing for updated software is indispensable before its releasing. State-of-the-art increasing number of researchers have been devoting to develop new techniques that can automatically generate high-coverage test suites and detect software errors introduced by patches. In this paper, we proposed a technique based on concolic execution to ensure the correctness and reliability of a patch. Our method generates test inputs to cover the changed lines of the patch and the relevant function by using a target-based search strategy which combines the selector based on the mapped address and the selector based on the priority. A prototype system called FCEP was implemented and evaluated with 5 C-programs. The experimental results demonstrated that our method reaches the new code introduced by patches quickly and achieves a high coverage.

Index Terms—concolic execution, patch testing, Search strategies

I. INTRODUCTION

The extensibility of software is both a blessing and a curse. On the one hand, one can easily add new functionality or patches to fix incorrect behavior. On the other hand, any software changes may introduce unexpected errors and security vulnerabilities, which are disasters for users and make users hesitate to update their software. As matter of fact, many people prefer to not upgrade their software to the latest version \cite{1, 2}, but rely on earlier versions which also usually contain serious errors and reduced functionality. Therefore, it is very necessary to perform a comprehensive test of the updated software. However, software testing is expensive and time-consuming as it involves writing a large numbers of manual test suites to validate various paths. This is a tedious process that requires an immense amount of work and a good understanding of the tested system. Some recent testing effort focuses on code that has changed from one version to the next \cite{3–7}.

Patches, as a typical form of software changes, ideally, should be comprehensively tested, but this level of testing is still far from being achieved in practice \cite{8}. State-of-the-art some studies \cite{6, 9, 10} test the patches based on concolic execution \cite{11, 12}, which is a program analysis technique that provides the ability to generate inputs to form high-coverage test suites. Concolic execution has proved to be a good choice to comprehensively test real software \cite{13, 14}, with its ability to systematically explore different program paths. Most of the work on concolic execution is focused on whole program tests, in which all parts of the program are treated equally. However, the number of execution paths in a program is usually exponential in the number of branches, exploring all possible execution paths is infeasible. That is, concolic execution faces challenges of path explosions \cite{11, 15}. An efficient search strategy of concolic execution is very important to overcome the path explosions challenge in testing patches. Despite recent progress, the studies still far from reaching the goal of fast and automatically generating test cases that contain code changes in the actual program.

In this paper, we have developed an automated testing method called FCEP based on concolic execution to ensure the correctness and reliability of patches, which takes advantages of both static and dynamic analysis to generate test inputs to cover the changed lines in patches. Instead of exploring all branches in the candidate list, FCEP searches priorities branches according to the distance between the uncovered statements and the targets (e.g. lines in patch). FCEP combines the exploration of the patch with the exploration of the function where the patch is located and dynamically adjusts the search target to conduct a more comprehensive test of the patch-related functions.

This paper mainly makes following contributions:

- FCEP ensures the reliability and security of software updates by using a target-based search strategy to test the patch and the relevance function quickly, which combines the selector based on the mapped address and the selector based on the priority, to test the patch and its relevant function quickly.
- The selector based on the mapped address reaches the patch quickly by calculating the shortest distance between candidate states and targets (e.g. lines in patch). The selector based on the priority comprehensively tests the relevant function of the patch as soon as possible by assigning priority to the related states.
- FCEP reduces the false negative by comprehensively
testing the relevant function of patches and modifying the CFG (Control Flow Graph) in real-time based on the results of concolic execution. For example, guiding the path search in dynamic analysis only according to the results of static analysis cannot cover relevant paths containing indirect jumps. 

- We performed experiments on 5 C-programs. FCEP covered more than 90% of the patch lines and found 34 out of 39 bugs in 5 tested software in the least amount of time. The experiments showed that FCEP can cover more lines of code in patches, and can quickly find bugs introduced by patches.

The rest of the paper is organized as follows: Section II introduces concolic execution and describes several representative search strategies found in the literature; Section III details our approach; Section IV shows the evaluation plan and the experimental results; Section V discusses related work, and Section VI concludes.

II. BACKGROUND

a) Concolic Execution: Concolic execution is an automatic test generation technique based on symbolic execution, a program analysis technique that can systematically explore paths through a program. The key idea behind symbolic execution is to run the program with symbolic values instead of concrete ones. Then, whenever an encountered branch is directly or indirectly dependent on the symbolic input, execution determines the feasibility of both sides of the branch, and creates two new independent symbolic states which are added to a worklist to follow each feasible side separately. Finally, whenever a path terminates or hits an error, the constraints on that path are solved to produce a concrete input that exercises the path. Since the number of execution paths in a program is usually exponential in the number of branches, exploring all possible execution paths is infeasible. To address this problem, concolic execution relies on the search heuristic to steer concolic execution in a way to maximize code coverage in a given limited time budget [11].

b) Search strategies: Since enumerating all paths of a program can be very expensive, in many software engineering projects related to testing and debugging, the search is prioritized by looking at the most promising paths first. Depth-first search (DFS) and breadth-first search (BFS) are the most common strategies. DFS expands a path as much as possible before backtracking to the deepest unexplored branch, while BFS expands all paths in parallel. DFS is often adopted when memory usage is at a premium. The breadth-first search (BFS) strategy traverses the execution tree according to a BFS order. The BFS strategy prefers branches that appear early in the execution paths, therefore generating new input vectors is easier because a smaller number of constraints will be involved for those branches. Hence, in spite of the higher memory pressure and of the long time required to complete the exploration of specific paths, some tools resort to BFS. In theory, both DFS and BFS strategies can cover all execution paths in the execution tree. However, as described in the previous section, real world programs have a nontrivial number of execution paths and neither strategy scales to even medium-sized programs [11], [12], [16]. Another popular strategy is random path selection [14], which has been refined in several variants.

III. SYSTEM DESIGN AND IMPLEMENTATION

In this section, we first show an overview of FCEP. We then explain the address mapping technology and the strategy of path selection.

A. Overview

Fig. 1 demonstrates an overview of FCEP. The inputs of our technique are: 1) the new version of program and the patch, and 2) the inputs selected from the test suite. The output is a set of inputs that trigger crash bugs or cover the code of patch.

On the one hand, FCEP uses the disassembler to generate the CFG and the CG (Call Graph) of the new version program. FCEP marks the position in the CFG for each target which is a line of the patch, and generates a list of function address ranges based on the CG. On the other hand, the executor creates a new state when it encounters a branch. And at the same time, the address finder searches the address of the next instruction in the generated CFG for both states. After mapping the new state to the address of the next instruction, FCEP puts the newly generated state into the candidate pool to wait for the next time selecting of the selector to complete concolic execution.

When the path entered with the initial value has executed, FCEP selects a new execution state by adopting a target-based search strategy in the candidate pool to perform concolic execution. Concolic exploration that focuses on target-based search provides inputs for a crashing path. Our search strategy infers the paths which are not covered by patches to avoid exploring large numbers of paths, and to direct the search towards the paths covered by patch.

Once there is a state that triggers the function where the patch is located, the target-based search strategy chooses the new state which falls into the state of the target function as much as possible. This is mainly because for the testing of patch, we believe that only covering the line of patch is far from meeting the testing requirement of ensuring the security of the patch. At the same time, FCEP uses the concolic execution to correct the paths which through indirect jumps or the function pointer calls in the statically generated CFG. Once indirect jumps or function pointer calls are encountered in the tested paths, FCEP will splice the related indirect jump blocks to find more paths.

B. Address Mapping

a) Generate CFG&CG: The first step of our analysis is determining the differences between the new program version and its previous version, (i.e. the patch). Theoretically, each line in the patch is a potential target to our FCEP. Whereas, many lines can be overlooked in practice because the source code contains many non-executable lines (e.g., declarations,
Fig. 1. A high-level overview of our execution.

C. The Target-based Search Strategy

a) Selector based on the mapped address: New states are generated where the conditional branch is located by executor, and FCEP put them into a candidate pool for subsequent selecting. The selector selects a new state from the candidate pool following a search strategy to continue the concolic execution when the execution of a path is finished. Since the search strategy usually affects the coverage of concolic execution, careful selection of the algorithm can help to achieve the desired goal.

Before the start of the test, FCEP has obtained the address range of each function from the CG, and has selected a path marked in the CFG which can execution from the function main to the selected core target. When the execution of the initial state is finished, FCEP determines whether the state newly generated hits the target function by the address of the next instruction, in other words, whether the address is within the range of the target function. If it hits, FCEP selects the path closest to the core target to continue running. If there is no state hits the target function, FCEP looks for the caller of the target function and confirm whether there is a state hits the caller. By analogy, FCEP selects the closest one among states which hit the caller to continue execution. Since it has been confirmed whether the new state previously generated hits the target function or its callers, FCEP gives priority to the new states derived from the current running state to confirm whether they fall into the target function.

As we all know, code addresses of a program are not completely continuous, but they are continuous in a same function of a program. Therefore, when selecting the closest state, FCEP calculates distance between the new state and the target using the formula $D_i = |state\_i addr - target\_addr|$. For the target function, the target is a line of patches determined before running the program, and for other functions in the call chain of the target function, the target is the line of the function call.

b) Selector based on the priority: When a new branch state is generated, FCEP determines whether its next instruction address hits the target function (that is, the function where the patch is located). If it hits, FCEP gives priority to the state. When a path hits the core target code (that is, one line of the
patch), FCEP continues to select the state that hits the target function to perform the concolic execution until there is no state hits the target function or the expected time runs out. In this step, FCEP only needs to execute the state with priority until all the states with priority have been executed.

After that, it continues to perform all the above steps for the next core target code. The purpose of running the states that hit target function is to test as much code near the patch as possible. Because for the security testing of patches, only paying attention to the path covered by the patch code itself is often unable to meet the requirements. The code near the patch is often associated with the patch, so it is necessary to test the function where the patch is located.

D. An Example of Search Strategy

Fig. 2 is an example of the search strategy, and it is a combination of CFG and CG. Assume that the address range of each function obtained from CG is \( FunA(0x00, 0x2F) \), \( FunB(0x40, 0x6F) \) and \( FunC(0x0A, 0xAF) \). The core target selected in the target function \( FunC \) is located at node \( n_{12} \), and a path sequence selected in the function call chain passing through the core target is \( \{ n_0, n_4, n_5, n_{10}, n_{12} \} \). FCEP determines the sequence of the function call node and its address as \( n_5(0x12) \). The execution tree is empty when the concolic execution starts. Suppose state_0 is the initial state and it corresponds to the sequence of \( \{ n_0, n_1, n_2, n_7, n_8 \} \) when FCEP runs the program with the initial inputs. At the same time, this state creates three new branch states \( \{ state_1, state_2, state_3 \} \) in the concolic execution. FCEP queries the address of the next instruction for the three new states in the CFG (that is, the address of the next block in the CFG) and puts them into the candidate pool. Assuming that the address of the next instruction of state_1 is 0x80, the address of state_2 is 0x06, and the address of state_3 is 0x4F.

When the state state_0 is executed, the selector selects a new state from the candidate pool. FCEP first determines whether the addresses of the next instruction in the three newly generated states hit the target function FunC. If there are some states in the range of the target function, FCEP selects a state closer to the core target node to continue the concolic execution. If there is no such state, the address funder queries whether there are states in the range of the caller function (e.g. the function \( FuncA \)). As shown in the Fig. 2, the three states do not hit the target function \( FunC \), but there are two states \( \{ state_1, state_2 \} \) hit the upper function of \( FunC \), that is, \( FuncA \). The node \( n_5(0x12) \) is the point that the \( FuncA \) calls the \( FuncC \). FCEP calculates all the distances between the states and the address of node \( n_5 \), that is, \( D_1 = 4, D_2 = 6 \), so it chooses the state_1 which has a smaller distance as the next execution state.

Assuming the execution path of the state_1 is \( \{ n_0, n_4, n_5, n_{10}, n_{11} \} \), it generates two new states \( \{ state_4, state_5 \} \), and their addresses are 0x1F and 0xAE. FCEP gives priority to judge whether the newly generated states hit the target function and finds the state_5 has located in the target function, so this state is selected as the next execution state. After that, FCEP continues to select the state derived from state_5 in the target function to complete the concolic execution.

E. Implementation

We implemented the proposed FCEP as a plugin of S2E [17] which is a general concolic execution framework. This plugin is mainly composed of three custom modules:

a) Automatic constructing CFG&CG: This module builds CFG and CG for the tested program and marks the selected target and path.

b) Address finder and mapping: This module finds the address of the next instruction in the CFG for the new state when the executor creates a branch state, and puts the mapped address into the candidate pool. When indirect jumps or function pointer calls is executed, this module dynamically corrects CFG and CG.

c) The target-based selector: This module uses the target-based search strategy to select a new state to continue concolic execution after a path is finished. This module combines the exploration of the patch with the exploration of the function where the patch is located to dynamically adjust the search target.

IV. Evaluation

We evaluated FCEP experimentally with real-world application binaries, answering the following research question:

- Effectiveness of generated heuristics: Can FCEP generate effective search heuristics? What is the coverage for patches?
- Bug detecting ability: Does FCEP generate effective search heuristics and how faster FCEP detect target bugs than the current concolic execution techniques?

We conducted all of the experiments on a computer running Ubuntu 18.04 64-bit, equipped with a 3.4 GHz Intel Core i7-6700 CPU and 24 GB of RAM. We evaluated FCEP with software patches from GNU Coreutils application suite. We only tested 8 programs that contain errors in the Coreutils application binaries, answering the following research question:
TABLE I
INFORMATION OF PATCHES AND BUGS

<table>
<thead>
<tr>
<th>Targets</th>
<th>Lines</th>
<th>Func.</th>
<th>Target bugs</th>
<th>Patches (LoC)</th>
<th>Patches (Func)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coreutils-6.10</td>
<td>4570</td>
<td>97</td>
<td>8</td>
<td>34</td>
<td>8</td>
</tr>
<tr>
<td>Grep-2.0</td>
<td>5956</td>
<td>132</td>
<td>6</td>
<td>53</td>
<td>6</td>
</tr>
<tr>
<td>Make-3.75</td>
<td>28715</td>
<td>555</td>
<td>10</td>
<td>109</td>
<td>10</td>
</tr>
<tr>
<td>Sed-1.17</td>
<td>4085</td>
<td>73</td>
<td>3</td>
<td>71</td>
<td>2</td>
</tr>
<tr>
<td>Vim-5.0</td>
<td>66209</td>
<td>1749</td>
<td>12</td>
<td>262</td>
<td>16</td>
</tr>
<tr>
<td>Sum</td>
<td>109535</td>
<td>2602</td>
<td>39</td>
<td>529</td>
<td>42</td>
</tr>
<tr>
<td>Average</td>
<td>21907</td>
<td>520</td>
<td>7.8</td>
<td>105.8</td>
<td>8.4</td>
</tr>
</tbody>
</table>

TABLE II
LINE COVERAGE FOR PATCHES OF EACH TARGET

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Coreutils</td>
<td>234</td>
<td>44.80%</td>
<td>69.71%</td>
<td>88.90%</td>
<td></td>
</tr>
<tr>
<td>Grep</td>
<td>353</td>
<td>41.61%</td>
<td>58.15%</td>
<td>94.67%</td>
<td></td>
</tr>
<tr>
<td>Make</td>
<td>909</td>
<td>53.27%</td>
<td>63.49%</td>
<td>89.07%</td>
<td></td>
</tr>
<tr>
<td>Sed</td>
<td>171</td>
<td>29.89%</td>
<td>55.36%</td>
<td>87.63%</td>
<td></td>
</tr>
<tr>
<td>Vim</td>
<td>1262</td>
<td>48.52%</td>
<td>70.04%</td>
<td>91.03%</td>
<td></td>
</tr>
<tr>
<td>Average</td>
<td>505.8</td>
<td>43.62%</td>
<td>63.35%</td>
<td>90.26%</td>
<td></td>
</tr>
</tbody>
</table>

test set, including: paste, pr, tac, mkdir, mknfio, mknod, ptx and seq. Furthermore, we collected real-world bugs (shown in TABLE I) from SIR [18] C programs which were fixed by the original developers from Dec 1996 to July 2018. TABLE I shows the detail of the 8 tools in Coreutils and the 4 software (Grep, Make, Sed and Vim).

A. Effectiveness

In order to determine the effectiveness of heuristics, we ran S2E, KATCH [6] and FCEP with the above test software and patches for 100 hours respectively. The results are displayed in TABLE II. The second column of the table is the total number of lines of the function where the patches are located. FCEP achieved an average line coverage of 90.26%, which is 1.42 (90.26/63.35) times larger than that of KATCH and 2.07 (90.26/43.62) times larger than that of S2E. Because FCEP not only tests the line of the patch itself, it also tests other codes in the function where the patches are located, so its coverage is much higher than other testing tools. Experiment shows that in the same time, FCEP can concentrate resources on comprehensive testing where the patches are located.

As a matter of fact, some patches are macro-defined code blocks which were not compiled in our compiled environment, so that some patches were not covered. Dynamic symbol execution for a macro-defined code blocks is a common problem for they may not be compiled. Some patches are referenced header files and newly defined variables. For newly defined variables, the location where the variable is referenced can be tested. New variable definitions and new header files do not cause problems because they are reflected in the code which is really changed and are tested by that code. In our experiments, the macro-defined code described above is excluded whereas the remained patches were covered.

B. Bug Detecting Ability

Coreutils-6.10 contains 8 vulnerabilities (paste, pr, tac, mkdir, mknfio, mknod, ptx and seq) and there are 6, 10, 3, and 12 bugs in Grep, Make, Sed, and Vim, respectively. TABLE III summarize the number of detected bugs and the time spending for the three methods. The third column in the table is the running time of each program. In particular, the sixth column of the table (FCEP*) lists the data obtained when the target patches is covered, and does not include the data of a comprehensive search for the function where the patch is located.

For a same software, FCEP successfully detected more bugs than the other two. KATCH found 21, S2E found only 12, whereas FCEP detected 34 over 39 bugs in total, showing much higher bug detect rate for patches. Notably, some bugs are not crash errors so that platform is difficult to detect these bugs without adding assertions to the code. So FCEP still missed 5 errors. FCEP found 34 bugs in 576 hours, while KATCH took 576 hours to find 21 bugs and S2E found 12 bugs. The time period required for FCEP to detect a bug is 16.94 (576/34) on average, whereas those for KATCH and S2E are 27.43 (576/21) and 48 (576/12) respectively. FCEP*, which only tests the line where the patch is located and does not fully test the corresponding function, the time for finding each bug is 19.86(576/29).

Compared with S2E, FCEP finds more bugs in the same time. This is because it can reach the code block where the patch is located more quickly and concentrate resources to test the location of the target to reduce the exploration of redundant paths. Compared with KATCH, FCEP can correct the CFG obtained by static analysis in real time to detect the corresponding paths which through indirect jumps or the function pointer calls. Compared with FCEP*, which only detects the patch code line, FCEP can find more bugs in the same time because it can perform a more comprehensive test on the function where the patch is located.

V. RELATED WORK

In recent years, there has been a lot of research on bug search in programs based on patches, but the technical methods used are also different. SPAIN [19] is a patch analysis framework to automatically learn the security patch patterns and vulnerability patterns, and identify them from the program binary executables. But SPAIN focus on patches in which only
one function is modified for one patch, but do not support patches where multiple functions are changed for one patch.

Based on derived operation semantic and constraint formula from patched differences, PVDF [20] computes the semantic of patches for privilege elevation vulnerabilities. This work is similar to SPAIN, but it assumes the availability of patches, and only focuses on one particular vulnerability type. Differently, SPAIN attempts to summarize patterns for different vulnerability types, and only requires the binary programs but not the patches.

Shadow symbolic execution [10] is a novel technique for generating inputs that trigger the new behaviors introduced by software patches. However, Shadow is not fully automatic, while many of the annotations added could be automated, manual assistance might still be needed.

Several heuristic-based approaches have been proposed to guide an execution toward a specific branch. KATCH [6] is a technique for patch testing that combines symbolic execution with several novel heuristics based on program analysis that effectively exploit the program structure and existing program inputs. Compared with manual testing, despite the increase in coverage and the bugs found, KATCH was still unable to cover most of the targets. Because it does not handle with the paths which through indirect jumps or function pointer calls.

VI. Conclusion

Software updates are easy to introduce bugs, so a full test of the software patch is indispensable, but extremely expensive and time costing. In this paper, we develop a method called FCEP to ensure the reliability and security of software updates by using a target-based search strategy to test the patch and the relevance function quickly, which search strategy combines the selector based on the mapped address and the selector based on the priority. In addition, FCEP reduces the false negative by comprehensively testing the relevant function of the patch and modifying the CFG in real-time based on the results of concolic execution.

Experiments initially show that FCEP can lead to significant improvements in reducing the number of path to explore and the time-cost to reach the patch-related code. So it can exclude uninteresting parts of code during analysis and focuses on those paths most relevant to the patches. At present, FCEP can only solve part of the problems of indirect jumps and function pointer calls. In the future, we will further study the automatic identification of them.

ACKNOWLEDGMENT

This work is supported by the strategic Priority Research Program of Chinese Academy of Sciences, Grant No.XDC02010400.

REFERENCES


Leveraging Compiler Optimization for Code Clone Detection

Shirish Singh
Dept. of Computer Science
Columbia University, USA
shirish@cs.columbia.edu

Harshit Singhal
Dept. of Computer Science & Engineering
The LNMIIT, India
18ucc159@lnmiit.ac.in

Bharavi Mishra
Dept. of Computer Science & Engineering
The LNMIIT, India
bharavi@lnmiit.ac.in

Abstract—Finding similar code in software systems can guide several software engineering tasks such as code maintenance, program understanding, and code reuse. Similar code detection has been actively studied in the past. In the paper, we propose a novel approach that leverages compiler optimizations to transform semantically similar code and detect similar programs. The key observation of our work is that the compiler optimizations can be used to smooth out source code level idiosyncrasies introduced by the developers, thus making the optimized programs, for the same task, similar in structure. The similarity in structure can then be used to classify the programs. We conducted experiments on the Google CodeJam dataset to demonstrate the effectiveness of our approach. The experimental results show that our technique can achieve up to 85% accuracy on the program classification task, which is an improvement of more than 25% over the source code level classification.

Index Terms—code clones, compiler optimization, reverse-engineering, code representation

I. INTRODUCTION

Code clone detection is an important problem for software maintenance and evolution. Several approaches have been studied for clone detection, which can be subdivided into two broad categories: a) Static analysis: extraction information from the code content [1] and b) Dynamic analysis: clone detection based run-time program behavior. [2]. Applications of code clone detection are manifold, such as code maintenance, program understanding, malware detection, and code reuse. In this work, we rely on compiler optimizations to classify the semantically similar programs. Compiler optimization is a sequence of transformations performed by the compiler on a program to produce a semantically equivalent binary that uses fewer system resources for its execution. Our study’s main idea relies on the hypothesis that the compiler optimizations can be used to remove any source code level idiosyncrasies introduced by the developers, thus making the optimized code, for the same task, structurally similar. This similarity can then be used to classify the programs.

A. Motivating Example

Code snippet 1 and 2 represent the multiplication functionality using two different methods: simple multiplication and multiplication by addition. After compiling the snippets with O3 compiler optimization and then decompiling the binaries through Ghidra, we observe that the resultant code is the same, shown in code snippet 3.

```c
int main(int num1, int num2){
    return num1 * num2;
}
```

Snippet 1: Simple Multiplication

```c
int main(int num1, int num2){
    int multiplication = 0;
    for(int i = 0; i < num2; i++){
        multiplication += num1;
    }
    return multiplication;
}
```

Snippet 2: Multiplication via repeated addition

```c
ulong main(int param_1,int param_2) {
    return (ulong)(uint)(param_1 * param_2);
}
```

Snippet 3: Ghidra Decompiled Code

As seen in the above code snippets, the main idea behind our study relies on the key observation that the compiler transforms the code for optimized performance, and two semantically similar codes can yield very similar (often overlapping) optimized binaries. Hence, we leverage these compiler optimizations to reduce the differences in the source code introduced by the developers. Figure 1 depicts the key idea of our work. Two semantically similar developer-written programs can have a large distance ($\Delta_1$) when represented as vectors. After compiling (with optimizations) and then decompiling, we observe that the effective distance between the decompiled programs ($\Delta_2$) reduces significantly to deem them similar. The net change ($\Delta_1 - \Delta_2$) in the distances between the original programs and the decompiled programs results in significant improvements in the clone detection task.

B. Contributions

In this paper, we answer three research questions:

- **RQ1**: Can compiler optimization be used to smooth out code level differences introduced by the developer?
- **RQ2**: Can the compiler optimized code be used to detect similarity? If yes, then which optimizations are optimal?
- **RQ3**: Can cross-optimization detect similar code?

The primary contributions of this paper are three-fold:
Fig. 1: $\Delta_1$ represents the distance between the vector representations of two semantically similar source code. $\Delta_2$ represents the distance between the vector representations of the decompiled binaries of the source code.

- We present a novel technique for code clone based on compiler optimizations. Our approach can also be adapted to detect similar binaries without source code availability.
- We study the impact of different compiler optimization levels on code clone detection. We also perform experiments to investigate the impact of cross-optimization on clone detection.

We conducted experiments on the Google Code Jam dataset from 2008 to demonstrate the effectiveness of our proposed approach. The experimental results show that our technique can achieve accuracy up to 85% on the classification task. We also study the effectiveness of cross compiler optimizations on the classification task. To the best of our knowledge, this is the first work exploring compiler optimized decompiled code for code clone detection tasks at the source code level. Our work is a general framework that can be adapted to solve other challenges such as malware detection, plagiarism detection, etc.

The remainder of the paper is organized as follows: Section II explains the background on code clones, compiler optimizations, and code embedding. Section III discusses the data-set used in our study. Section IV describes the proposed framework. Section V presents the results of our study, followed by a discussion section and a section on related work. Section VIII discusses the limitations of our work. Finally, section IX discusses the conclusion and future work.

II. BACKGROUND

In this section, we discuss code clones, compiler optimizations, Ghidra, and code representation through code2vec.

1) Code Clones: Code clones are similar pieces/fragments of code that are either syntactically or behaviorally similar. In practice, programmers often use clones via copy/paste to support rapid software development. For a given code snippet, there can be several types of clones. Four types of code clones have been widely studied in literature [3], [4]: Type-1 (textual similarity), Type-2 (lexical, or token-based, similarity), Type-3 (syntactic similarity), and Type-4 (semantic similarity).

2) GCC Compiler Optimizations: GNU Compiler Collection (GCC) is the GNU compiler project which supports several high-level languages, such as C and C++. One core function of the compiler is to optimize the code for performance. Code optimization has several benefits; it allows reduced resource consumption, resulting in faster running machine code and lesser memory usage. The optimization is performed by doing transformations (optimizations) that can only be done at the assembly (machine) level for the target hardware. The GCC optimizer supports six pre-defined optimization levels: -O1, -O2, -O3, -Ofast, -Og, and -Os [5]. In this work, we utilize -O1, -O2, -O3 optimization levels.

3) Ghidra: Ghidra is a free, open-source reverse engineering tool developed by National Security Agency (NSA) [6]. It is a comprehensive and expandable framework covering the complete workflow of binary analysis. Ghidra is often used for the decompilation of executable binaries, and in this study, we use Ghidra’s command-line analysis tool to reverse-engineer the decompiled code of compiler optimized C/C++ code binaries.

4) Code2vec: Code2vec [7] is a neural network architecture based on attention architecture for representing snippets of code as continuous distributed vectors or code embeddings. Originally trained on Java, Code2vec converts the source code into a set of paths using the code’s underlying Abstract Syntax Tree (AST) and learns how to combine these paths using an attention mechanism. Code2vec then represents each function as a fixed-length code vector which is used to represent the different features of that function. Method embeddings generated by code2vec serve as a base for a large variety of applications and analyses such as author attribution, bug detection, and so on. It has been shown that the generated embeddings can be aggregated using several aggregation methods such as max, min, sum, mean, median, and standard deviation to obtain embeddings at a file-level [8]. We utilize median aggregation method to represent each program.

III. DATASET

Code Embedding Dataset: Since the original code2vec model is trained on Java language, we trained a new code2vec model on C and C++ programs from top 1000 Github repositories. Because of memory limitations, we excluded the Linux repository.

Experiment Dataset: Google CodeJam (GCI) is a yearly programming competition hosted by Google. In our study we used GCJ dataset from 2008 [9] provided on Github [10]. The competition has several rounds, each containing several problems to be solved by the participants worldwide. The diverse characteristics of the participant pool introduces diversity in the submissions for any given programming task. Participants are allowed to submit their programs in any language of their choice. In this study, however, we only
consider C and C++ programs because of compiler restrictions. The GCJ dataset can be further sub-divided into two types of programs: accepted solutions and non-accepted submissions. In this study, because of ground truth availability, we only use the accepted solutions. In our study, the submissions from 2008 GCJ were used to extract the code embedding, train, and test the classifiers. The 2008 data contained 8,524 solutions written in C/C++ with disproportionate distribution across different problems. For consistency, we consider six programming tasks with about 200 randomly sampled submissions. The total size of the dataset was 1,423. We further split the data into training and test set, containing 1,280 and 143 submissions, respectively. The programs were then compiled with three different optimization levels (see section II-4) and then decompiled using Ghidra [6] (see section II-3).

IV. SYSTEM OVERVIEW

Our approach involves four steps: a) Code compilation using compiler optimizations, b) Code decompilation, c) Generation of code embedding, and d) Classifying the embedding in to clusters of similar code. Figure 2 shows the high level overview of our pipeline. Given a C/C++ program, we first compile the binary using one of the optimization flags to generate a binary executable. Then we use Ghidra to reverse engineer the binary to retrieve the source code. The generated source code is then fed to the code2vec model to retrieve the code embedding for the program. The embedding is used to train a model to classify the programs. In this section, we discuss all the steps in further detail.

A. Compiling Binaries

We use the GCC compiler to generate the GSJ dataset’s source programs’ binaries in the first step. For every program, we generate three binaries corresponding to three optimization levels: O1, O2, and O3. These binaries are then decompiled using Ghidra reverse-engineering tool to get the source code. All binaries were compiled for x64 architecture.

B. Ghidra for decompilation

Our study uses Ghidra’s command-line analysis tool to reverse-engineer the decompiled version of compiler optimized C/C++ code binaries. The study uses command-line analysis, also known as headless-analysis, since work requires several files to decompile at once, so it is feasible to use command-line analysis. We first import all binaries and then perform analysis to decompile them. The decompiled code is saved in separate files to generate code vector representations.

C. Code Embedding

The main component of getting code vectors from code2vec is path extraction. Code2vec first constructs the AST (Abstract Syntax Tree). Then the syntactic path between AST leaves are extracted, which form the path-context. Each path and leaf-values of a path-context is mapped to its corresponding real-valued vector representation, or its embedding. Then, the three vectors of each context are concatenated to a single vector that represents that path-context.

It is important to note that code2vec generated code embedding for methods as opposed to programs or files. Since we wanted to study program level similarity rather than function level similarity, we had to generate a single code embedding for a single file that might contain multiple functions. As shown in prior research [8], we can get the file level embedding by aggregating the set of method level embedding. The aggregation method is applied column-wise. The base aggregation functions used are max, min, mean and median. A combination of aggregation methods can also be considered. In our study, median aggregation worked best, so we constructed the program level embedding using median aggregation.

1) Model Training: Since the base code2vec model is trained on Java language, we trained a new code2vec model on C and C++ programs from top 1000 Github repositories. Code2vec model generation is a two-step process: pre-processing and model training. For pre-processing step, we used the pre-processing script provided by code2vec_c [11]. We set the maximum leaf node to be processed in the given method to 320. We pre-processed all C programs in the repositories; however, some files that did not match the maximum leaf node size training criteria were removed. The remaining 1.2 million programs were then used to train the code2vec model. The model was trained for 40 epochs.

2) Feature Vector Extraction: For getting the code embedding for the classification task, we used the newly trained code2vec model. We captured code-vector corresponding to each function in the program. Since we wanted to study program level similarity rather than function level similarity, we had to generate a single code embedding for a single file that might contain multiple functions. To generate one vector to represent a given program, we used median aggregation function following prior research [8], which showed that we can get the file/program level embedding by aggregating the set of method level embedding. These program level embedding were then used to train and test the classification models.

V. EVALUATION AND RESULTS

A. Experimental Setting

We use five off-the-shelf machine learning algorithms and one customized DNN to train our models and demonstrate the effectiveness of the proposed work. We trained five off-the-shelf machine learning algorithms for classification tasks: Random Forest (RF), K-Nearest Neighbour (KNN), Support
TABLE I: Performance comparison table shows the accuracy and F1-score achieved by each model.

<table>
<thead>
<tr>
<th>Model</th>
<th>Source</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
<th>O1,O2,O3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Acc.</td>
<td>F1</td>
<td>Acc.</td>
<td>F1</td>
<td>Acc.</td>
</tr>
<tr>
<td>RF</td>
<td>53.14%</td>
<td>52.00%</td>
<td>49.65%</td>
<td>50.71%</td>
<td>56.64%</td>
</tr>
<tr>
<td>KNN</td>
<td>76.95%</td>
<td>76.95%</td>
<td>72.02%</td>
<td>71.56%</td>
<td>83.21%</td>
</tr>
<tr>
<td>LR</td>
<td>59.08%</td>
<td>59.44%</td>
<td>83.21%</td>
<td>74.03%</td>
<td>76.49%</td>
</tr>
<tr>
<td>F1</td>
<td>83.21%</td>
<td>74.03%</td>
<td>76.49%</td>
<td>83.21%</td>
<td>53.14%</td>
</tr>
<tr>
<td>DT</td>
<td>74.12%</td>
<td>11.88%</td>
<td>66.43%</td>
<td>84.96%</td>
<td>71.12%</td>
</tr>
<tr>
<td>SVM</td>
<td>68.53%</td>
<td>69.23%</td>
<td>86.13%</td>
<td>84.61%</td>
<td>84.61%</td>
</tr>
<tr>
<td>DNN</td>
<td>84.94%</td>
<td>42.03%</td>
<td>84.61%</td>
<td>84.61%</td>
<td>86.01%</td>
</tr>
</tbody>
</table>

Vector Machines (SVM), Logistic Regression (LR), and Decision Tree (DT). We also trained a DNN. The DNN consists of an input layer (containing 384 neurons corresponding to the 384 code2vec features), one hidden layer (384 neurons and ReLu activation), and a softmax output layer (6 classes corresponding to 6 programming tasks).

We ran our training and testing scripts on a Dell XPS 8930, with Intel i5-9600K 6-core 64GB RAM, running Ubuntu 18.04 and Python 3.7.3. To evaluate the performance of the model, we use four metrics, namely, accuracy, precision, recall, and F1-score. Because of space limitations, we only report accuracy and F1-score.

**B. Code Classification**

We selected six programming tasks from the 2008 Google Code Jam dataset, each having about 200 data points. The dataset’s total size was 1,423 programs, which was split into training and test set containing 1,280 and 143 programs, respectively. We then extracted code vectors corresponding to each program and aggregated the vectors to get a program-level representation. The process was repeated for decompiled code for different optimization levels O1, O2, O3. Ultimately, we compiled four embedding datasets for the classification task: original programs, O1 optimized, O2 optimized, and O3 optimized programs. In addition to the four datasets, we also merged all the optimized program embeddings (O1, O2, and O3) to test the performance of the models.

TABLE II: Performance of cross optimizations (Accuracy)

<table>
<thead>
<tr>
<th>Models</th>
<th>Source</th>
<th>O1</th>
<th>O2</th>
<th>O3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train Set</td>
<td>Test Set</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source</td>
<td>O1</td>
<td>O2</td>
<td>O3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>51.04%</td>
<td>83.21%</td>
<td>84.61%</td>
</tr>
<tr>
<td>RF</td>
<td></td>
<td>22.37%</td>
<td>65.03%</td>
<td>68.53%</td>
</tr>
<tr>
<td>KNN</td>
<td></td>
<td>15.38%</td>
<td>72.72%</td>
<td>74.82%</td>
</tr>
<tr>
<td>LR</td>
<td></td>
<td>18.88%</td>
<td>68.53%</td>
<td>83.91%</td>
</tr>
<tr>
<td>F1</td>
<td></td>
<td>51.04%</td>
<td>83.21%</td>
<td>84.61%</td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td>42.03%</td>
<td>83.21%</td>
<td>84.61%</td>
</tr>
<tr>
<td>SVM</td>
<td></td>
<td>65.03%</td>
<td>83.21%</td>
<td>84.61%</td>
</tr>
<tr>
<td>DNN</td>
<td></td>
<td>51.35%</td>
<td>78.56%</td>
<td>83.91%</td>
</tr>
</tbody>
</table>

**Fig. 3: Performance comparison chart compares the accuracy of each model on the optimization datasets.**

We trained the models on the code vector representation of the programs. Since we had six different classes of programs in the dataset, we trained multi-class classifiers. Table I summarizes the results of the models. We observe the classification accuracy is highest in the models trained on O2 optimized programs. Furthermore, it can be seen that using the best models, the classifier can correctly classify up to 84.61% of the decompiled programs vs. only 58.74% of the source code. We also observe that DNN and SVM models perform similarly. The models collectively trained on O1, O2, and O3 optimizations outperform other models trained on single optimizations, with significant margins.

Figure 3 summarizes the accuracy of all the models. We can observe that the accuracy is sub-optimal in the case of the developer written program (depicted as ‘Source’); however, the accuracy significantly increases if we apply a compiler optimization. This increase in the accuracy is owed to the transformations performed by the compiler on the source program, which results in similar binaries being constructed from semantically similar programs written by different developers.

**Fig. 4: Comparison Between Different Algorithms**

We trained the models on the code vector representation of the programs. Since we had six different classes of programs in the dataset, we trained multi-class classifiers. Table I summarizes the results of the models. We observe the classification accuracy is highest in the models trained on O2 optimized programs. Furthermore, it can be seen that using the best models, the classifier can correctly classify up to 84.61% of the decompiled programs vs. only 58.74% of the source code. We also observe that DNN and SVM models perform similarly. The models collectively trained on O1, O2, and O3 optimizations outperform other models trained on single optimizations, with significant margins.

Table II summarizes the results of the experiment in terms of accuracy. It can be observed that the models perform the best when tested on the same set of data it was trained on. This observation implies that the compiler transformations, under different optimization levels, produce binaries that are somewhat different from each other such that the code embedding cannot capture the similarities. Moreover, we see a unique pattern in the performance of the models trained on O1 and O3 optimized programs. For the model trained on O1 optimization, we see that the performance of the model is better on O2 dataset as compared with O3. Similarly, the performance of the model trained on O3 optimization is better on O2 as compared to O1. This pattern suggests some similarities between close levels of optimizations. The model trained on the original source programs exhibits the same pattern. As we move away from the original program, from O1 to O2 to O3, we observe a degradation of performance.

Figure 4 depicts the t-SNE plots of the code embedding for each dataset. It can be observed that source code is harder to separate, but compiler-optimized decompiled program representation shows significant improvement and allows the data points to separate. The level of separation increases as we increase the optimization level from O1 to O2. However, the separation decreases slightly in O3.
Fig. 4: t-SNE of code vectors for different optimization levels. Different colors represent different problems of the GCJ dataset. We can observe that optimization O2 segregates the data with a clear distinction between the clusters.

VI. DISCUSSION

RQ1: Can compiler optimization be used to smooth out source code level differences introduced by the developer?

Through the code classification study, we observed that compiler optimized code has significantly higher accuracy when compared to the original developer written source code. Since the code2vec representation relies on the AST of a given method, the compiler optimizations transform semantically similar high-level developer written code to programs with similarly structured ASTs. We also observed in the motivating example, that two semantically similar code snippets can often result in similar binary.

RQ2: Can the compiler optimized code be used to detect similarity? If yes, then which optimizations are optimal?

Through the experiments, we demonstrated that the classifier trained on compiler optimized programs embedding can out perform classifier training on the original source code embedding. In our study, we also observed that O2 optimization performs the best among all optimizations. We hypothesize that this phenomenon occurs because several O3 optimization flags transforms the loops of the program, thereby changing the AST of the method, which in turn impacts the code vector embedding of code2vec model.

RQ3: Can cross-optimization detect similar code?

We performed a study to learn about the impact of different optimizations on each other. We trained a DNN model on each optimization level and tested against the other optimizations. Table II summarizes the performance on cross compiler optimizations. We observe that each model learns to classify the programs with the same optimization they were trained upon. However, the models fail to adopt and perform sub-optimally on other optimizations. This outcome can be partially be attributed to vector embedding used to represent the code. We hypothesize that using a more comprehensive code representation can lead to improvement in classification task on cross compiler optimization (see section VIII).

RQ3: Cross compiler optimizations are not effective in detecting similar code.

VII. RELATED WORK

Code similarity has been extensively studied in literature [12], particularly type-4 (semantically similar) code clones. Through a user study, researchers showed that functionally similar code exists in practice [13]. While static token based approaches such as SourcererCC[14] and CCFinder[15] have been studied, advances in computing has paved the path for two other approaches to code clone detection. We first outline machine learning approaches based on static features of the code, followed by dynamic approaches.

A. ML for Code Clone Detection

Deep learning has also been applied for detecting code clones [16], [17], [18], [19]. Researchers used both structure or identifiers to detect all four types of code clones [16]. Their technique relied on a novel code representation scheme: the terms in code fragments were mapped to vector representations such that terms used in similar ways map to similar vectors. Then the model learns discriminating features for code fragments at different levels of granularity. DeepSim is another approach that measures code functional similarity [17] by encoding control flow and data flow graphs into a semantic matrix. Another similar approach, HOLMES [18] (that relies on CFG and DFG), performs semantic code clone detection using program dependency graphs and graph neural networks by leveraging the structured, syntactic, and semantic information of the source code. FCCA [19] uses hybrid code representation by combining unstructured (code as sequential tokens) and structured (ASTs and CFGs) information of the code. Authors then train a deep-learning model with attention. Asm2vec [20] is a binary clone detection system that uses vector representation of assembly functions to detect clones.
B. Dynamic Analysis for Code Clone Detection

Tajima et al. [21] proposed to detect functionally similar code for newly created methods that do not have test cases. Authors first extract interface information and PDG from methods. Then this information is used for similarity detection. Li et al. [22] proposed a technique based on automatic test case generation to search semantically equivalent API methods by running the generated test cases. They consider two methods to be similar if the methods generate the same output on each of the generated test cases. Mathew et al. [2] proposed SLACC, a cross-language clone detection based on runtime behavior. It uses function I/O to cluster code based on its behavior. Authors generate 256 inputs per function to find similarity. Compared to dynamic techniques, our work is lightweight since we do not need to run the programs.

VIII. THREATS AND LIMITATIONS

Our approach relies on Code2vec embedding that utilizes ASTs of the program to generate the vector representation. For Type-I (textual similarity), Type-II (lexical similarity), and Type-III (syntactic similarity) clones, code2vec produces significantly similar ASTs because of similarity in syntactic structure. However, Type-IV code clones are only behaviorally similar; they have different syntactic structures. Hence, the underlying ASTs of Type-IV clones are significantly different, leading to different code2vec vector representation. Moreover, code2vec has been shown to rely heavily on variable names for prediction, causing it to be fooled by typos or adversarial attacks [8]. Our code2vec model was trained on the source code, and using an obfuscated version of the training data can potentially improve the performance. Moreover, there are several techniques in the literature to generate code embedding that utilize call graphs, ASTs, and other data from the code [23]. In this work, our framework relies on code2vec embedding. It is possible that other representations, such as Asm2vec [20], yield better results. We leave this to future work.

IX. CONCLUSION

In this paper, we propose compiler optimization based code clone detection technique. Our approach relies on the compiler to smooth out the differences in the source code introduced by the developer. We observed that O2 optimization yields the best performance (84.61% accuracy and 84.96% F1-score) for the classification task among O1, O2, and O3. Our proposed approach yields an improvement of more than 25% accuracy over the source code based representation. Furthermore, we investigated the utility of cross compiler optimization for classification problem. Our results suggest that the optimizations yield significantly different binaries making it difficult for the model to learn optimally. In the future, we plan to study program representation to accommodate cross compiler optimization and improve classification performance.

REFERENCES


Abstract

With the spread of smartphones, the importance of automated testing of mobile applications has increased. However, many current approaches are inadequate, as they are not able to test functions that are available only on hard-to-reach GUI, which is a screen that can be reached only through a specific sequence of input events. To solve this problem, there has been an increase in testing research based on reinforcement learning, specifically Q-learning. Each research uses different learning targets and reward function. Testing research has also been done using Deep Q-Network, which extends reinforcement learning in a “deep” way. Although each work has conducted their own evaluation, it is not clear how the combination of learning algorithm, learning target, and reward function affects the result. To bridge this gap, we have conducted an empirical study comparing eight possible combinations. Our study found that the combination of Deep Q-Network as the learning algorithm, component as the learning target, and GUI change ratio as the reward function had the highest test quality in terms of code coverage.

Keywords: Software testing, Q-learning, Android application

1. Introduction

The proliferation of smartphones has led to an increase in mobile applications. As with conventional software, testing is an essential part of the development process of mobile applications. Thus, research on automated testing is being actively conducted. Unfortunately its performance is still not adequate. A major reason for this is that GUI screens which can be reached only through a specific sequence of input events are not easily reached. As a result, functions on those screens are often not tested.

In the past several years, reinforcement learning (specifically Q-learning), as well as a “deep” version of Q-learning (Deep Q-Network), has been applied to testing in order to overcome this issue [6] [4] [1] [5] [8] [9]. Reinforcement learning is a type of machine learning, where an agent adapts to the environment through trial and error. Unlike supervised learning, there is no teacher who explicitly indicates the correct output for the state input. Instead, the agent learns using reward.

Although each work has their own unique points, the basics are the same. The Q-learning agent interacts with the Android application to gradually build up a model and then creates test cases based on the model. The agent searches for the best input solution that allows it to test as many features as possible. Q-learning based testing has been able to achieve higher code coverage than random testing.

Still, there are some differences that need to be noted. First, we have already stated above that there is Q-learning as well as a “deep” version of Q-learning. Second, what the agent learns (i.e., learning target) differs; most work targeted events, while one targeted components. Finally, the function used to calculate the reward differs; some were based on the GUI change ratio, while others were based on “Optimistic Initial Value Method.” Table 1 summarizes the differences. Thus, we see at least three factors that need to be considered when applying reinforcement learning to testing, specifically learning algorithm, learning target, and reward function.

Although each work has conducted their own evaluation, it is not clear how the three factors affect testing. In order to bridge this gap, we have conducted an empirical study that compares the eight combinations that can be made from the three factors. Table 2 shows the eight combinations that we target in our study.

In the rest of this paper, we first describe the three fac-
Table 2. Eight Combinations based on the Three Factors of Learning Algorithms, Learning Targets and Reward Functions

<table>
<thead>
<tr>
<th>ID</th>
<th>Related Work</th>
<th>Learning Algorithm</th>
<th>Learning Target</th>
<th>Reward Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[1], [4]</td>
<td>Q Learning</td>
<td>Event</td>
<td>Optimistic Initial Value Method</td>
</tr>
<tr>
<td>2</td>
<td>[6], [5], [8]</td>
<td>Q Learning</td>
<td>Event</td>
<td>GUI Change Ratio</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Q Learning</td>
<td>Component</td>
<td>Optimistic Initial Value Method</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Q Learning</td>
<td>Component</td>
<td>GUI Change Ratio</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>DQN</td>
<td>Event</td>
<td>Optimistic Initial Value Method</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>DQN</td>
<td>Event</td>
<td>GUI Change Ratio</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>DQN</td>
<td>Component</td>
<td>Optimistic Initial Value Method</td>
</tr>
<tr>
<td>8</td>
<td>[9]</td>
<td>DQN</td>
<td>Component</td>
<td>GUI Change Ratio</td>
</tr>
</tbody>
</table>

Factors that we focus on: learning algorithm, learning target, and reward function. Section 3 describes the design of our empirical study. Section 4 discusses the results of our study. Section 5 discusses threats to validity. Section 6 makes concluding remarks.

2. Factors in Q-learning based testing

This section describes the three factors that we focus on: learning algorithm, learning target, and reward function.

2.1. Learning algorithm

Q Learning. Q-learning is a learning algorithm for reinforcement learning that has been used in the work of Mariani et al. [6], Korogulu et al. [5], Adamo et al. [1], Esparcia et al. [4], and Vuong et al. [8]. There are two methods of reinforcement learning: one based on value functions and the other based on strategy search. Q-learning is an algorithm based on value functions. The value function estimates how good it is for an agent to perform an action in a given state. The criterion for the estimation is the expected future reward, called the cumulative reward. Since the future reward depends on which action the agent will take, the number of values is defined according to a specific strategy. We define the value function \(Q^\pi(s, a)\) as follows:

\[
Q^\pi(s, a) = E[R \mid s, a, \pi] \tag{1}
\]

The value function returns the cumulative reward that can be achieved by performing a sequence of actions that starts from \(s\) with action \(a\), and then following the policy \(\pi\) from the succeeding state.

The optimal Q function \(Q^*\) returns the maximum cumulative reward that can be obtained from a given pair of state and action.

\[
Q^* (s_t, a_t) = \max_{\pi} \sum_{t=0}^{T-1} \gamma^t r_t \mid s = s_t, a = a_t, \pi \tag{2}
\]

If the optimal Q value \(Q^* (s_{t+1}, a_{t+1})\) for the next step is known, then the optimal strategy is to take the action that maximizes \(r + \gamma Q^* (s_{t+1}, a_{t+1})\). \(r\) is the immediate reward for the current step. \(Q^*\) satisfies the Bellman equation.

\[
Q^* (s_t, a_t) = r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \tag{3}
\]

\(\gamma\) is the discount rate, which is a value between 0 and 1. The discount rate determines whether to give more weight to immediate or cumulative rewards; closer to 0 means more immediate rewards and closer to 1 means more cumulative rewards.

The optimal strategy \(\pi^*\) is to take the action with the largest Q value \(Q^*\). The Q-learning algorithm iteratively calculates the value of the Q function based on equation (3). First, the Q-function is initialized with a default value. Each time the agent performs an action to go from state \(s_t\) to state \(s_{t+1}\) and receives a reward \(r_t\), the Q function is updated as follows:

\[
Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a)) \tag{4}
\]

\(\alpha\) is the learning rate, which is a value between 0 and 1. It determines how much the Q-value is to be updated.

On the surface, choosing the action with the largest Q value seems to be favorable, but there is a need to balance the trade-off between exploration and exploitation [7]. \(\epsilon\)-greedy approach is often used in reinforcement learning. Simply, a random action is selected with probability \(\epsilon\), or the action with the highest Q-value is selected with probability \(1 - \epsilon\).

Deep Q-Network. This is the learning algorithm used in the work of Vuong et al. [9]. The strength of neural networks lies in their ability to learn from low-dimensional feature representations and their ability to approximate complex functions. Using the approximation properties of neural networks, it is possible to approximate the optimal strategy \(\pi^*\) and the optimal value function \(Q^*\). The extension of reinforcement learning with neural networks is deep reinforcement learning. One of the best known methods for deep reinforcement learning is Deep Q-Network (DQN).
The optimal value function $Q$ can be obtained by a neural network by using the weights $\theta$.

$$Q(s_t, a_t, \theta) \approx Q^*(s_t, a_t) \quad (5)$$

Training is done by adjusting the weights $\theta_t$ at iteration $i$ so that the mean square error of equation (3) becomes small. The right term in equation (3) is replaced by the following:

$$r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}, \theta^\top_t) \quad (6)$$

2.2. Learning target

**Event.** Mariani et al. [6], Adamo et al. [1], Esparcia et al. [4] and Vuong et al. [8] targeted events for learning. GUI testing tools usually interact with the application under test by sending events to GUI components. For example, a click (event) is sent to a button (component) that transitions to the next page.

**Component.** Vuong et al. [9] targeted components for learning. The goal of reinforcement learning algorithms is to search for hard-to-reach application features in a way that reveals them. They tried to achieve this goal by considering the semantics of components and by making components as the learning target.

2.3. Reward function

**Optimistic Initial Value Method.** The optimistic initial value method is a reward function used in the work of Adamo et al. [1] and Esparcia et al. [4]. It is commonly used because it is simple to implement and effective for simple problems. However, so far no theoretical guarantees have been given, and in practice, it is not efficient because many iterations are required before the correct value propagates and overrides the optimistic value. As shown in equation (7), we make the function such that the reward for unexplored actions is maximized, so that all actions are explored exhaustively.

$$R(s_t, a_t) = \frac{1}{f(s_t, a_t)} \quad (7)$$

where $s_t$ is the state at step $t$, and $a_t$ is the action at step $t$. $R(s_t, a_t)$ is the reward for taking action $a_t$ in state $s_t$. $f(s_t, a_t)$ is the number of times action $a_t$ is taken in state $s_t$.

**GUI Change Ratio.** GUI Change Ratio is a reward function used in the work of Mariani et al. [6] and Vuong et al. [8] [9]. As shown in equation (8), by considering the percentage of GUI changes, the reward is determined so that new features can be explored.

$$R(s_t, a_t, s_{t+1}) = \frac{|s_{t+1} \setminus s_t|}{|s_{t+1}|} \quad (8)$$

where $s_t$ is the state at step $t$, and $a_t$ is the action at step $t$. $R(s_t, a_t, s_{t+1})$ is the reward for the transition to state $s_{t+1}$ as a result of taking action $a_t$ in state $s_t$. The right term in Equation (8) is a ratio that indicates how much the number of GUIs is changed when the agent transitions from one state $s_t$ to the next $s_{t+1}$.

3. Experiment Design

3.1. Overview

We conducted a comparative study of the eight combinations that are shown in Table 2, and aim to answer the following four research questions:

- **RQ1:** Which reinforcement learning algorithm is better: Q-learning or Deep Q-Network?
- **RQ2:** Which learning target is better: events or components?
- **RQ3:** Which reward function is better: optimistic initial value method or GUI change ratio?
- **RQ4:** Which combination will give the highest test quality?

The implementation of the eight combinations were done by extending existing implementations. For combinations which are based on Q-learning, we extended ClassicQ, which was originally implemented in [8]. For combinations which are based on Deep Q-Network, we extended QDroid which was originally implemented in [9].

We investigated the code coverage for twelve Android applications to evaluate test quality. Although not perfect, code coverage is often used to check the quality of test. We used Androtest [2], an automated test tool evaluation framework, for obtaining code coverage. We measured class coverage, method coverage, block coverage, and line coverage. Two-hour tests were conducted five times for each of the twelve applications under test, and the average code coverage was calculated.

3.2. Parameter Settings

Two important parameters in Q-learning is discount rate $\gamma$ and learning rate $\alpha$. We took into account the parameter values used in previous work, and also conducted some trial-and-error executions of our tool. Based on this, we chose the values for these parameters to be $\gamma = 0.9$ and $\alpha = 1.0$.

Another important parameter is $\epsilon$. As with [8], the initial value of $\epsilon$ is set to 1 (i.e., always randomly choose an action), and continually decreased it until $\epsilon = 0.5$.

3.3. Target Applications

The applications to be tested as benchmarks are the datasets used in Vuong et al. [9]. These applications are the ones included in Androtest. The dataset consists of twelve Android applications, as shown in Table 3.
4. Experiment Results and Discussion

Table 4 shows the average values of class coverage, method coverage, block coverage, and line coverage for each of the eight combinations. Table 5 shows the average method coverage values for each of the combination and each of the application.

For both Tables 4 and 5, the result in the row with the best coverage value is in bold font. So, for example, in Table 4, ID8 had the best class coverage at 62.58%, while ID2 had the best method coverage for the application AnyMemo at 38.8%. For the results of each application, we only show method coverage and not the other three coverages due to space issues, but the tendency was the same.

We now discuss each research question. For each research question, we first discuss based on Table 4 which gives the overall results, and then discuss based on Table 5 at the application level.

4.1. RQ1: Which reinforcement learning algorithm is better: Q-learning or Deep Q-Network?

Since we had learning target and reward function as factors, in order to compare Q-learning and Deep Q-Network, we compared each of the pair (ID1, ID5), (ID2, ID6), (ID3, ID7), and (ID4, ID8). When we look at the results of each of these pairs in Table 4, in all cases the combinations using Deep Q-Network had the better results. We also conducted statistical analysis, but we did not obtain a significant difference in each of the pair.

When we look at the results for each application in Table 5, we can see that this depends on each application. ID8 was better than (or the same as) ID4 for all applications. But for the three other pairs, about half had Q-learning better, and about half had Deep Q-Network better.

**RQ1 Answer**: Overall, Deep Q-Network was found to be better than Q-learning, but the difference was not statistically significant. When looking at each application, there was not a clear cut tendency for one over the other except for ID8, which was better than (or the same as) ID4 for all applications.

4.2. RQ2: Which learning target is better: events or components?

Similar to RQ1, we compared each of the pair (ID1, ID3), (ID2, ID4), (ID5, ID7), and (ID6, ID8). In most cases in Table 4, the combination using components had a better result. Only the class coverage and line coverage for the pair (ID5, ID7) had events with the better results. We also conducted statistical analysis, and we found that for the pair (ID6, ID8), ID8 (component) was significantly different (better) than ID6 (event). The difference in other pairs were not statistically significant.

When comparing for each application (Table 5), again there was not a clear cut tendency towards either event or component, except for ID8. ID8 was better than (or the same as) ID6 in eleven out of twelve applications.

**RQ2 Answer**: Component was found to be better than event in most cases, but there was one pair where the difference was found to be statistically significant; ID8 (component) was found to be better than ID6 (event). This was also seen at the application level.

4.3. RQ3: Which reward function is better: optimistic initial value method or GUI change ratio?

We compared each of the pairs (ID1, ID2), (ID3, ID4), (ID5, ID6), and (ID7, ID8). Looking at the overall results in Table 4, for the pairs using Q-learning, i.e., (ID1, ID2) and
Table 4. Overall Results: Average

<table>
<thead>
<tr>
<th>ID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>54.10</td>
<td>53.24</td>
<td>57.38</td>
<td>57.05</td>
<td>58.42</td>
<td>60.03</td>
<td>57.89</td>
<td><strong>62.58</strong></td>
</tr>
<tr>
<td>Method</td>
<td>44.80</td>
<td>44.37</td>
<td>47.02</td>
<td>46.41</td>
<td>47.83</td>
<td>48.45</td>
<td>47.88</td>
<td><strong>52.98</strong></td>
</tr>
<tr>
<td>Block</td>
<td>40.82</td>
<td>39.90</td>
<td>43.08</td>
<td>41.75</td>
<td>43.25</td>
<td>43.75</td>
<td>43.84</td>
<td><strong>47.95</strong></td>
</tr>
<tr>
<td>Line</td>
<td>39.98</td>
<td>39.25</td>
<td>42.33</td>
<td>41.40</td>
<td>43.18</td>
<td>43.48</td>
<td>42.98</td>
<td><strong>47.75</strong></td>
</tr>
</tbody>
</table>

Table 5. Results of each Application: Method Coverage

<table>
<thead>
<tr>
<th>App Name</th>
<th>ID1</th>
<th>ID2</th>
<th>ID3</th>
<th>ID4</th>
<th>ID5</th>
<th>ID6</th>
<th>ID7</th>
<th>ID8</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnyMemo</td>
<td>36.0</td>
<td><strong>38.8</strong></td>
<td>26.8</td>
<td>26.0</td>
<td>28.2</td>
<td>28.2</td>
<td>31.4</td>
<td>31.8</td>
</tr>
<tr>
<td>My Expenses</td>
<td>29.0</td>
<td>35.2</td>
<td>35.0</td>
<td>39.8</td>
<td>45.6</td>
<td>50.4</td>
<td>30.6</td>
<td><strong>64.4</strong></td>
</tr>
<tr>
<td>Who has my stuffs</td>
<td><strong>81.4</strong></td>
<td>69.8</td>
<td>75.2</td>
<td>76.2</td>
<td>76.4</td>
<td>76.4</td>
<td>79.8</td>
<td>80.2</td>
</tr>
<tr>
<td>Tippy Tipper</td>
<td>52.8</td>
<td><strong>54.8</strong></td>
<td>54.4</td>
<td>53.8</td>
<td>54.2</td>
<td>54.0</td>
<td>50.4</td>
<td>54.6</td>
</tr>
<tr>
<td>Munch Life</td>
<td><strong>51.2</strong></td>
<td>48.0</td>
<td>48.0</td>
<td>48.0</td>
<td>48.0</td>
<td>48.0</td>
<td>48.0</td>
<td>48.0</td>
</tr>
<tr>
<td>Mini Note Viewer</td>
<td>42.2</td>
<td>39.6</td>
<td>32.6</td>
<td>37.4</td>
<td>38.0</td>
<td>38.0</td>
<td>41.0</td>
<td><strong>48.2</strong></td>
</tr>
<tr>
<td>Mileage</td>
<td>34.4</td>
<td>35.3</td>
<td>27.2</td>
<td>26.4</td>
<td>26.8</td>
<td>27.6</td>
<td>33.6</td>
<td><strong>35.5</strong></td>
</tr>
<tr>
<td>Multi SMS sender</td>
<td>37.4</td>
<td>37.2</td>
<td>38.0</td>
<td>36.8</td>
<td>37.0</td>
<td>37.0</td>
<td>35.0</td>
<td>37.2</td>
</tr>
<tr>
<td>Hot Death</td>
<td>16.4</td>
<td>17.0</td>
<td>59.2</td>
<td>60.6</td>
<td>59.8</td>
<td>61.2</td>
<td>59.0</td>
<td><strong>64.2</strong></td>
</tr>
<tr>
<td>Random Music Player</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
<td>54.0</td>
</tr>
<tr>
<td>Dalvik Explorer</td>
<td><strong>80.6</strong></td>
<td>73.8</td>
<td>76.6</td>
<td>64.2</td>
<td>65.4</td>
<td>65.4</td>
<td>77.5</td>
<td>78.6</td>
</tr>
<tr>
<td>Weight Chart</td>
<td>22.2</td>
<td>29.0</td>
<td>37.2</td>
<td>33.8</td>
<td>40.6</td>
<td><strong>41.2</strong></td>
<td>34.2</td>
<td>39.0</td>
</tr>
</tbody>
</table>

(ID3, ID4), optimistic initial value method had better results, but it was not statistically significant. But for the pairs using Deep Q-Network, i.e., (ID5, ID6) and (ID7, ID8), it was the opposite, i.e., GUI change ratio had better results. However, in these cases also, the difference was not statistically significant.

When we look more closely at the application level in Table 5, we can see that for the pairs using Deep Q-Network, change ratio had the same or better results in all but one case, where in Tippy Tipper ID5 was better than ID6 by just 0.2%, which should be considered as negligible. For the pairs using Q-learning, there was little difference between the two reward functions.

RQ3 Answer: There was no statistically significant difference. But there was a tendency for optimistic initial value method to be better for Q-learning, and change ratio to be better for Deep Q-Network.

4.4. RQ4: Which combination will give the highest test quality?

From Table 4, we can see that ID8 (Deep Q-Network, component, GUI change ratio) was found to have the best results for all types of coverage. We also checked the statistical difference between ID8 and each of the other combinations. Except for the pair (ID1, ID8), the difference was statistically significant for all other combinations.

This can also be seen in Table 5. ID1 had three apps with the best results, while ID8 had four. ID8 was better than ID1, in terms of number of apps with the best results, although not by much.

We also note that ID2 had two apps and ID3 and ID6 had one app each with the best result. However, for these four apps, we can see that the difference with the other combinations were not that large. When compared with ID8, the difference in these four apps ranged from 0.2% (Tippy Tipper) to 7.0% (AnyMemo).

RQ4 Answer: The combination of Deep Q-Network, component and GUI change ratio was found to be the best combination.

4.5. Further Discussion

In Table 5, when comparing the results for ID1 and ID8, we can also see that the results for ID8 was more stable. The lowest and second lowest results for ID8 were 31.8% (AnyMemo) and 35.5% (Mileage), while the two lowest results for ID1 was 16.4% (Hot Death) and 22.2% (Weight Chart). Note though that although Mileage was the second worst result for ID8, it was still the best result among all eight combinations for Mileage.

Although we focused on the three factors of learning algorithm, learning target and reward function, we must not forget that other parts still need work, especially being able to generate “meaningful” strings. For example, in Table 5, the results of Random Music Player for all combinations
was 54%. We manually checked the results, and found that it wasn’t just the method coverage value itself that was the same; the methods that were covered were also the same. This was because one of the functions in Random Music Player requires the input of a URL. However, none of the eight combinations were able to generate a meaningful URL, and thus all functions (methods) that can be used after entering a URL could not be tested.

5. Threats to Validity

Internal Validity. We limited each execution to two hours. Since reinforcement learning is an approach that learns while executing, there are two possible issues. First, it may be possible that the coverage would continue to increase if the execution time was longer. Second, the shape of the coverage curve may differ between execution, i.e., some executions may cover more code quickly while others may not be as quick. Third, there may be differences between each execution. Although we cannot completely negate these possibilities, we tried to minimize these as much as possible by taking the average of five executions for each combination and application.

Another threat to internal validity is the parameters of reinforcement learning. The execution results will vary depending on the parameters values, specifically discount rate $\gamma$, learning rate $\alpha$, and $\epsilon$-Greedy value $\epsilon$. To mitigate this threat, we selected parameter values such as discount rate and learning rate based on empirical analysis reported in previous studies, as well as some trial-and-error execution of the combinations.

External Validity. In this study, we targeted twelve Android applications. In terms of sampling bias, it is possible that completely different results could be obtained if the current test were conducted on different applications.

Construct Validity. We used code coverage to assess how good an app was tested. Using code coverage for this purpose has long been considered to be controversial [3]. Thus, code coverage may not be perfect for comparison. But code coverage is used in many testing papers, and we believe that it is adequate enough as one way to compare testing approaches.

6 Conclusions and Future Work

We conducted an empirical comparative study of Android application testing focusing on the three factors of learning algorithms, learning targets, and reward functions. We implemented eight combinations based on these three factors and executed them on twelve applications, and measured code coverage. We compared and discussed the eight combinations based on four research questions. We found that the combination of Deep Q-Network, component and GUI change ratio was the best combination (RQ4). For the other research questions, there was little statistical significance, although we did discuss some trends.

Future work includes investigating other Android applications to eliminate external validity. Also, executing for more than two hours needs to be considered. Finally, as was discussed in subsection 4.5, one major issue that needs to be solved regardless of the three factors is being able to generate “meaningful” strings when necessary.

References


MACA: A Residual Network with Multi-Attention and Core Attributes for Code Search

Lian Gu, Zihui Wang, Jiaxin Liu, Yating Zhang, Dong Yang, Wei Dong
College of Computer Science, National University of Defense Technology, Changsha, China
{gulian, wangzihui98, liujiaxin18, zhangyating18}@nudt.edu.cn, yangdong5002@163.com, wdong@nudt.edu.cn

Abstract—Code search technique has gradually become a key skill to accelerate software development. However, the current deep learning methods only use the encoded results and ignore the original content of the code. Besides, the feature expression of the code is too single, which makes the model’s understanding insufficient. And the last problem is the lack of separate processing of core attributes, which will cause the model to lack differentiated learning of the attributes with different importance. Therefore, we propose a residual network based on Multi-Attention, so that the model can not only retain the original content of the code but also allow the code to perform a large number of combined learning in different aspects to obtain differentiated features. Then we treat three core attributes and specific implementation of the code differently so that the model can pay extra attention to the core attributes. We use 158,201 Java code-comment pairs for training. In our experimental results, our model is 9.5% higher than the existing method on the indicator of MRR and 12% higher on the SuccessRate@1.

Index Terms—code search, deep learning, residual network, Multi-Attention

I. INTRODUCTION

The code search field can be divided into two historical development stages. The first stage is based on information retrieval methods, and its main strategy is keyword matching; the second stage is based on deep learning methods, and its main strategy is to build a neural network model from query sentences to codes to bridge the semantic gap between the two.

On the one hand, the method based on information retrieval, because it relied on keywords, leads to limited correctness, and may not match codes that are highly similar to the query sentence due to different keywords. On the other hand, the current deep learning method has three problems. The first is that the model only uses the encoded result of the code and discards the original content, which will make the model lose a certain degree of stability. The second is that the model’s distributed representation of code features is too single, which makes the model unable to fully learn the code features. The last problem is that these models lack separate processing of core attributes, which will allow code elements of different importance to be treated equally, so that core attributes cannot be expressed as they should in the modeling process. Therefore, we propose a residual network structure based on Multi-Attention. This structure can not only retain the original content of the code but also allows the code to learn different aspects of the combination through a large number of attention mechanisms [1], so as to obtain a more comprehensive representation of the code. Finally, we extracted the three core attributes of the method name, return type, and parameter list separately from the original code content. Then let the distributed representation of these three attributes concatenate the existing coding content so that the model can pay extra attention to the important features of the code.

Our experimental results show that our method is higher than other deep learning methods in both SuccessRate and MRR indicators. This shows that our method effectively improves the performance of code search.

Our contributions are as follows:

• We find that inputting the core attributes of the code separately into the model increases the performance of the model;
• We conduct a comparative experiment on the presence or absence of Multi-Attention and find that Multi-Attention can effectively mine the potential information of code and comments;
• We also set up an experiment, discarding one of the attributes each time, and find that the method name helps the model the most.

II. RELATED WORK

The field of code search has always been a popular research content in academia and industry. It has gone through two research phases, one is realized by information retrieval technology, and the other is realized by deep learning.

At the stage where information retrieval technology is the main method, the code search method has already made many achievements. One of the typical methods is CodeBroker [2], which uses annotations to calculate similarity. Besides, Apache releases Lucene [3], which is an open-source full-text search engine toolkit, which can perform full-text indexing and search with high search efficiency. In addition to the full-text search engine Lucene, there are many code search engines based on information retrieval, such as Codase [4], Koders [5], Krugle [6].

Sachdev et al. set up an experiment to compare the effects of traditional information retrieval methods and deep learning methods on code search tasks [7]. The results show that methods based on deep learning can express more precise
semantics and achieve better performance. In recent years, a large number of code search methods based on deep learning have emerged. A typical method is NCS [7], which trains the code and query sentence at the same time to obtain a fastText [8] embedding, and then calculates the weight of the code according to IF-IDF [9] to obtain the final code representation vector, and directly averages the embeddings of the query sentence to obtain its final representation. Cambronero et al. improves based on NCS and proposes UNIF [10], which trains a fastText embedding for the code and query sentence respectively, and this embedding can be fine-tuned in the later training phase, and the weighting of the code is changed from TF-IDF to the attention mechanism. Gu et al. split code into three parts according to the characteristics: method name, API sequence, and code tokens [11]. And they train an embedding for each of the three, which encodes the method name and API sequence through the Recurrent Neural Network, and encodes the code tokens through the Multi-Layer Perceptron. Haldar et al. propose a multi-perspective architecture, which calculates the similarity by capturing both global and local similarities [12]. Mou et al. embed codes by a tree-based convolutional neural network [13]. Chen et al. model code and natural language by training two VAEs [14].

III. METHOD

Our model is a twin tower model based on the deep structured semantic model(DSSM) framework [15]. The twin tower model divides the input into two independent terminals, one for codes and another for comments, and then processes the two different inputs separately. Among them, each end is composed of three layers. The first layer is the input layer, also called the embedding layer. Its function is to process the input into a numeric vector. The second layer is the presentation layer, also called the coding layer. Its function is to process the input vector into a single vector that can represent the entire input. The third layer is the matching layer. Its purpose is to score the similarity of the two final representations. The higher the score, the more similar the two inputs.

A. Input Layer

In the input layer, we usually choose some classic word vector representations. We choose Word2Vec [16] to learn word vectors for code and comments respectively. On the code side, in addition to the coding code itself, we also extracted the three attributes of the return value, method name, and parameter list; on the comment side, in addition to the coding comment itself, we also extracted the commented verbs and nouns. We believe that these separately extracted features can better help the model to express.

B. Presentation Layer

The presentation layer is the core of the entire model. Its role is to encode a collection of word vectors representing code and comments into a single word vector, which represents the entire code or comment.

From the framework of the presentation layer, both the code side and the comment side are two residual network structures. On the one hand, the model needs to learn new information from the original content, and on the other hand, it also needs to retain the original information to a certain
extent. Therefore, we use Multi-Attention to generate new content and then concatenate the original content. This is the first residual structure that learns new content by itself through the model. In the second stage, the model also needs to retain the existing content given artificially. On the code side, these contents are the return type, parameter list, and method name; on the comment side, these contents are the verbs and nouns that appear in the comment.

Because the attention mechanism is only an understanding of one aspect of the code, we have performed multiple attention calculations on the code, and we call this process Multi-Attention. The attention mechanism of each time is calculated as follows:

$$a_{i,k} = \frac{\exp(a_k \cdot e_i^T)}{\sum_{i=1}^n \exp(a_k \cdot e_i^T)}$$ (1)

Among them, $a_{i,k}$ is the weight of each $e_i$ vector, and $a_k$ is the attention weight coefficient. The target combined vector can be calculated as follows:

$$v_k = \sum_{i=1}^n a_{i,k}e_i$$ (2)

where $v_k$ is the k-th vector of the target vectors.

Finally, the word vector representing the entire code is cascaded to the new vector generated by Multi-Attention and is input to the encoding stage as detailed information together. When summing up the embedding of the four parts, we directly obtain the final code vector representation by averaging.

On the comment side, we first extract the verbs and nouns in the comment sentence through the Natural Language Toolkit (NLTK) and input the word vectors of the two separately into the final representation. The encoding of the entire sentence is consistent with the code side. First, additional information about the comment is obtained through Multi-Attention, then these vectors are concatenated to the original vector, and finally, the average is taken.

C. Match Layer

The presentation layer has coded the code and the comment into a vector respectively, and the function of the matching layer is to score the similarity of the two vectors representing the code segment and the natural language comment. For two vectors with equal dimensions, we generally use cosine similarity for calculation. The higher the cosine similarity score, the closer the two vectors are. The calculation formula of cosine similarity is as follows:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$ (3)

Among them, A and B respectively represent a vector, $A_i$ represents the i-th element in the A vector, and $B_i$ represents the i-th element in the B vector.

### IV. Experiment

A. Dataset

a) Data Collection: We first obtain Java projects with a stat greater than or equal to 10 on GitHub through the crawler. Then we parse each Java file in the project through the AST parser of Java Development Tools (JDT) to get information such as comment, method name, return type, parameter list, and method body. Among them, each function corresponds to a piece of data. We finally get 158,201 pieces of data, and then randomly selected 500 pieces of data as the test set.

b) Preprocessing: Because the naming convention in Java follows the camel case principle, and the actual semantics is a single word in the variable name instead of the entire, so we also de-camelize the function name and variable name according to the regular expression and keep the content Words with precise semantics. At the same time, to be more stable in the subsequent training of word vectors, we convert all words to lowercase.

c) Data set training: In the training phase, our training set is a triple which consists of the following parts: a code segment, a natural language description, and an integer tag. The value of this tag is either 1 or 0. 1 means that the code segment and the natural language description are the data in the original data set. And 0 means that this code segment does not match the natural language description, which is generated by random negative sampling. In our experimental data, the number of our negative samples is equal to the number of original samples.

B. Experimental Setup

In our experiment, our data set is trained for 10 epochs, and the batch size is set to 100. In the setting of the count of Multi-Attention, we find that 30% of the number of original embedding vector sets is the best. If the original vector set has 100 vectors, then 30 vectors will be generated after the Multi-Attention. The code segment and the Word2Vec vector described by natural language are trained separately, and the dimension of the word vector of both is 128. Our model is implemented on the TensorFlow framework, and the optimizer selected during training is Adam.

Our similarity is calculated by cosine similarity. For a training data triple $<\text{code}, \text{description}, \text{flag}>$, our loss function is defined as follows:

$$\mathcal{L}(\theta) = \frac{\sum_{i=1}^n (\cos(a_i, b_i) - \text{flag}_i)^2}{N}$$ (4)

Where N represents the number of training samples, $\cos(\cdot)$ is the cosine similarity mentioned above, the value of the flag is 0 or 1, 0 means randomly sampled data, 1 means original sample data.

C. Evaluation Index

In the test indicators, we adopt the commonly used SuccessRate@k and MRR. MRR is the average value of the inverse of the ranking of all test data in the sample, and SuccessRate@k is the ratio of the number of all samples ranked before K.
D. Baseline

We chose UNIF, a classic model among the code search models. The model is very lightweight. On the code side, UNIF transmits the token vector of the entire code to the encoder in the form of a bag of words. The encoder completes the final representation of the code through the attention mechanism; on the comment side, UNIF directly averages the word vectors of all words to get the final representation of the comment. Because the model is relatively simple, the actual effect of each sub-module of our model can be observed more clearly in the experiment.

E. Results

We set up two comparative experiments, one of which is used to prove the effectiveness of the module we designed, and the other experiment is used to compare the degree of improvement of each artificially extracted core attribute on the model effect. The first experiment is to prove the effectiveness of manually extracting core attributes and Multi-Attention separately:

<table>
<thead>
<tr>
<th>Model</th>
<th>SR@1</th>
<th>SR@5</th>
<th>SR@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACA(Multi-Attention)</td>
<td>0.57</td>
<td>0.754</td>
<td>0.8</td>
<td>0.652</td>
</tr>
<tr>
<td>MACA(Core Attributes)</td>
<td>0.608</td>
<td>0.79</td>
<td>0.85</td>
<td>0.691</td>
</tr>
<tr>
<td>MACA(MA+CA)</td>
<td>0.656</td>
<td>0.808</td>
<td>0.868</td>
<td>0.727</td>
</tr>
</tbody>
</table>

It can be seen from the experimental data from table I that our model performs better than the UNIF model. In terms of all indicators, our model is 10% higher than UNIF as a whole. In particular, on the SuccessRate@1 indicator, our model effect has increased by 12%, and on the MRR indicator, our model effect has increased by 9.5%. Which shows that our model is very accurate in calculating the similarity between comments and code.

Then we remove the core attributes and Multi-Attention respectively. We find that the effect of removing these core attributes is worse than removing Multi-Attention, but the effect of the two is still better than UNIF. After removing the core attributes, MRR dropped by 7.5%; after removing Multi-Attention, MRR dropped by 3.6%. Therefore, for the entire model, human input of core attributes can greatly improve the model’s effectiveness.

<table>
<thead>
<tr>
<th>Model</th>
<th>SR@1</th>
<th>SR@5</th>
<th>SR@10</th>
<th>MRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACA</td>
<td>0.656</td>
<td>0.808</td>
<td>0.868</td>
<td>0.727</td>
</tr>
<tr>
<td>MACA-w/o.MethodName</td>
<td>0.598</td>
<td>0.774</td>
<td>0.816</td>
<td>0.676</td>
</tr>
<tr>
<td>MACA-w/o.ReturnType</td>
<td>0.644</td>
<td>0.806</td>
<td>0.856</td>
<td>0.718</td>
</tr>
<tr>
<td>MACA-w/o.Parameter</td>
<td>0.64</td>
<td>0.788</td>
<td>0.848</td>
<td>0.711</td>
</tr>
<tr>
<td>MACA-w/o.Verbs</td>
<td>0.65</td>
<td>0.802</td>
<td>0.858</td>
<td>0.721</td>
</tr>
<tr>
<td>MACA-w/o.Nouns</td>
<td>0.652</td>
<td>0.796</td>
<td>0.854</td>
<td>0.719</td>
</tr>
</tbody>
</table>

We finally set another comparative experiment, which was to remove one of the core attributes to see how the model’s effect declined. It can be seen from table II that when the core attribute of the method name is removed, the MRR index drops the most, up to 5.1%. When the parameter list is removed, MRR drops by 1.6%. Among other core attributes, the decline is minimal. This shows that it is necessary to input the two core attributes of the method name and parameter list to the model.

V. Conclusion

We propose a model based on the residual network, which deliberately encodes the return type, parameter list, and method name of the code on the basis of encoding the entire code. The core attributes are entered separately to enhance the modeling of the code by these attributes. In addition, in the process of encoding the entire code and natural language description, our model generates fresh vectors through Multi-Attention, and these new vectors again input to the presentation layer in the form of residuals as important content. Our experiments show that adding these important contents separately will enhance the effect of the model.

References


Dynamically Detecting Invariants for Automatic Testing PLC Programs

Zeyu Lu†, Xia Mao†, Yanhong Huang‡§*, Jianqi Shi‡‡, Yang Yang†
†National Trusted Embedded Software Engineering Technology Research Center
East China Normal University, Shanghai, China
‡Hardware/software Co-Design Technology and Application Engineering Research Center, Shanghai, China
§Shanghai Key Laboratory of Trustworthy Computing, Shanghai, China
Email: {zeyu.lu, xia.mao, yang.yang}@ntsec.ecnu.edu.cn, {yhhuang, jqshi}@sei.ecnu.edu.cn

Abstract—Since programmable logic controllers (PLCs) control safety-critical infrastructures, examining the PLC software satisfies the high-reliability specifications necessary to ensure the safeness of PLCs. However, prior works have limitations in finding defects in the PLC source code. Static verification techniques suffer from notable false positives without capturing runtime behavior. The symbolic execution and conformance testing technique captures the relations of inputs and outputs. It is not sufficient to consider only the data constraints as the PLC operates in real-time. In this paper, we propose a novel approach in the detection of the runtime behavior of PLC programs with incorporated time constraints. This testing approach automatically finds implementation errors in PLC programs by mining invariants from runtime traces. As the existing tools mine only data or time invariants which are inadequate to test PLC programs, our approach focuses on the interplay of data and time invariants. Dynamically detected data-time invariants are then checked with the safety specifications. We evaluate the usefulness of our approach in a real-life case. The experimental results show that the proposed approach can find errors in PLC programs effectively.

Index Terms—programmable logic controller, program invariant, real-time system, interplay

I. INTRODUCTION

Industrial Control Systems (ICS) have been used widely in many safety-critical domains, such as smart power grids, nuclear power plants, and transportation systems. These areas play an essential role in modern society. A programmable logic controller (PLC) is an industrial computer that is capable of being programmed to perform control functions. Now, PLCs are the most widely-used industrial process control technology. Since the PLC software controls safety-critical infrastructures, its inherent defects may have severe consequences, such as financial and property losses.

To ensure control logic safety, many previous studies [1]–[4] statically verified PLC programs to discover security bugs. These studies converted the PLC program to a model checker’s input language, such as NuSMV [5] or UPPAAL [6], and the model checker automatically checked whether the program satisfied the given formal specification. However, these approaches suffer from notable false positives because the error checking was performed statically without the program having been executed. These approaches found violation paths that could not be executed at runtime. Besides, these approaches are at the abstract model level, which is more suitable for checking design defects rather than implementation errors.

In recent years, several studies [7], [8] used symbolic execution and concolic testing to automatically generate test cases for PLC programs. In addition, Provost et al. [9] used conformance testing to test whether the execution code of the PLC conforms to the specification. Although these studies can perform white box testing by using symbolic execution or black-box testing based on the conformance testing technique, these techniques focus only on the relationship of inputs and outputs. It is sufficient to capture data value relations for typical software without considering the timing of the system under test. However, it is inadequate for the testing of time-constrained software since PLC is a real-time system.

To address the difficulties in detecting the runtime behavior of PLC programs with incorporated time constraints, we propose a novel approach to test PLC programs. Our approach is at the code level. It mines program invariants from runtime traces of the program under test. A program invariant, or property, is a condition that holds at a given point. Mining invariants from runtime traces eases the notable spurious warnings result from the static analyzers. Our technique further mines time invariants, considering PLCs operate in real-time.

However, existing dynamic invariant detection tools extract one-dimensional models, such as data or time, without capturing the interplay of them. Both data and time models are useful in that PLCs to operate in real-time and the data values of variables also express events that occur in the system. In this paper, we focus on the interplay of data and time invariants to find source code defects in PLC programs. Different program invariants capture different runtime behaviors during execution. Data Invariant expresses the range of values assigned to the variables and the relation of values of different variables. Time Invariant describes the time boundary of events that occur in a system. Data-Time Invariant illustrates the timing constraints of data invariants.

Our automatic testing approach is threefold. First, we instrument the program under test with the input and output relations. Second, we mine data-time invariants to observe the time performance of the implemented control logic. Third,
the dynamically detected data-time invariants are compared with the manually crafted specification, which expresses the expected behavior of the PLC software. Besides, we mutate the existing test suites to obtain adequate test cases to improve the quality of invariants derived. Once the PLC program has been tested and satisfies the safety requirements, it will be downloaded to real PLCs.

We evaluate the effectiveness of our proposed approach on a real-life case, i.e., cosmetic packing process. The experimental results show that the dynamically discovered invariants are efficient to help test PLC programs. We have found a latent error that is not easy to discover by using the existing approach such as symbolic execution in our implementation of the cosmetic packing system. To the best of our knowledge, we are the first to mine data-time invariants of programs dynamically in the context of ICS. In summary, this paper makes the following contributions:

1) We propose a testing technique that uses dynamically detected invariants to discover implementation errors in PLC programs at the code level.

2) We propose methods to derive data-time invariants from execution traces of PLC programs. The data-time invariants express the runtime behavior of the PLC programs more accurately.

3) We perform static analysis of the program under test to derive data invariants specifically tailored to PLC programs.

The rest of the paper is organized as follows. Section II provides some background and a motivational example. Section III illustrates our approach. Section IV presents the evaluation results of our approach. Section V discusses related work. Section VI concludes the paper.

II. PRELIMINARY AND MOTIVATION

A. Programmable Logic Controller

The program of PLC is executed continuously and each execution is called a scan cycle. Each scan cycle of a PLC consists of the following three processes: (1) sensor measurements are read to input variables, (2) the control commands are computed based on sensor values and the control logic, and (3) the control commands are sent to actuators which change the physical processes. The PLC sits in the closed-loop to perform control functions.

PLC Programming Languages. IEC 61131-3 standard is the third part of the IEC 61131 standard which provides standards to programmable controllers. There are five programming languages included in the IEC 61131-3 standard: namely, ST, IL, LD, FBD, and SFC. The five programming languages share many common elements and can be transformed with each other [10]. In this paper, we focus on the Structured Text (ST) language which offers a flexible way of expressing complex functionality.

B. Motivation Example

We present an example of a flashing light to illustrate that the runtime properties detected by existing tools cannot accurately reflect the behavior of a program. It is essential to mine the combined data and time properties of PLC programs to deal explicitly with time measures.

The expected behavior of the PLC program is as follows: if the weight on the conveyor exceeds a pre-defined constant value for some time, a warning light PL1 will begin to flash. Additionally, the solenoid is de-energized. The light PL1 flashes with the time interval of one second. If there is no anomalous situation, the PL1 light will keep off state. If the start button is pressed, the motor will start one second later.

The snippet of the implemented ST program is shown in Figure 1. The program has three input variables, i.e., Weight, OnOff, and Start, and three output variables, i.e., PL1, Motor, and Solenoid. The output variable PL1 is connected to a lamp. The state of PL1 can be judged from whether the lamp lights up (line 7). The delayed start of the motor is controlled by a timer (lines 8-9). The state of Solenoid is affected by whether the weight exceeds the preset value or not (line 10).

To test whether the implemented program satisfies the expected behavior, we first use Daikon [11] to detect runtime invariants. After running Daikon, the derived data invariant of variable PL1 is "Weight >= MaxValue" \(\Rightarrow\) PL1 one of \{false, true\}. Besides, if we specify the MaxValue as 26 and use the approach in [12] to find Weight equals 26 in the execution trace manually, Perfume [13] will infer the invariant Weight=26 \(\rightarrow\) PL1 \{1s, 2s\}. However, Daikon can only detect invariants of data value relations. It cannot express the time boundary of the relations. The invariant mined only by Perfume cannot describe the relationship of predicate Weight \(\Rightarrow\) MaxValue with variable PL1.

In this example, the derived data-time invariant by applying our approach is Weight \(\Rightarrow\) MaxValue \(\rightarrow\) PL1 \{1s, 2s\} when the scan cycle of the program is 50 milliseconds. This invariant expresses that, if the weight exceeds a preset value, the light PL1 will stay off for at least 1 second and will then turn on for 1 second. Besides, it shows that the runtime behavior is correct in the existence of timers, which delays the operation for one second.

III. APPROACH

A. Overview

Figure 2 shows the core workflow of our approach. The process of the test workflow can be summarized as follows:
We first use an open-source PLC compiler named matiec [14] to compile the PLC program to ANSI C code. The C code is semantically equivalent to PLC programs, and there are existing works [7], [12] using the matiec compiler to test PLC programs. For the translated C program, we instrument it using static analysis to leverage Daikon mining data invariants. Daikon’s front-end produces a trace file that records the values of the variables, and Daikon dynamically detects data invariants based on the trace file. After data invariants have been derived, we combine Perfume to capture the time constraints of data invariants. The detected data-time invariants are checked with the manually crafted specifications. The mismatches between the detected invariants and the specifications indicate that the PLC program contains errors. In addition, to improve the quality of detected invariants, we mutate test cases generated by symbolic execution to produce adequate test cases.

B. Instrument ANSI C Programs

To leverage Daikon to detect invariants tailored to PLC programs, we instrument C code with additional program points, i.e., instrument code with dummy procedures [11]. The dummy procedures do not affect the normal execution of the program and can help Daikon detect invariants concerning specific variables. The arguments of a dummy procedure include the variables of which we want Daikon to detect invariants and the timestamp that records the calling time of the dummy procedure.

For C or Java program, Daikon infers invariants at the granularity of function. However, the PLC program is not composed of functions. There are three types of program organization units (POUs) in PLC, i.e., Program, Function Block, and Function. Each PLC program may be comprised of several POUs. If the invariants are detected at the level of POU, the granularity is too coarse as there may be hundreds of lines of code in one POU. By contrast, if the invariants are generated at the statement granularity, then the number of detected invariants is huge and some of them do not make sense. We detect the invariants at the granularity of several correlated statements. The granularity is coarser compared to the statement level and finer compared to the POU level. These statements start with the one that input variables lie in and end with the one that the output variable sits. The change of the inputs impacts the output. We perform static analysis to obtain all input-output relations and instrument the ANSI C code with these relations.

We first generate the control-flow graph (CFG) of the PLC program. Every statement in the program is represented as a node in the CFG. The control-dependence and data-dependence are computed from the CFG. We then obtain the program dependence graph (PDG) based on the computed control and data dependence. We start from each node containing input variables and then traverse the PDG by depth-first search (DFS). If the traverse reaches a node that contains the definition of an output variable, the statements both in the node and the start node will be output. The traversal will output all of the input-output node pairs in the form of “if-then” relations. The statement in the input node is the condition and the output variable is the return variable. We pass the variables in the two nodes as arguments to the dummy procedure and add the “if-then” relations in the dummy procedure body. For example, the dummy procedure body of the two nodes “Weight >= MaxValue” and “PL1:= Flash I. FlashOut” in the motivation example is:

```
if (Weight >= MaxValue)
return PL1;
return PL1;
```

Specifically, there exists a case where a node contains both input and output variables. In this case, we simply pass the output variable in the node as arguments to the dummy procedure.

C. Invariant Types

Perfume mines property types based on the execution log and formalizes the mined properties using timed propositional temporal logic (TPTL) [15]. TPTL is a real-time specification language for the specification of real-time systems.

By default, Perfume mines seven property types, and we focus on four of them:

\[ \square x. (p \rightarrow (\diamond y. (q \land y - x \leq t))) \]  
\[ \square x. (p \rightarrow (\diamond y. (q \land y - x \geq t))) \]  
\[ \forall x. (p \rightarrow (\diamond y. (q \land y - x \geq t))) \]  
\[ \square x. (p \rightarrow (\diamond y. (q \land y - x \leq t))) \]

For the motivation example in Figure 1, the TPTL formula of the detected invariant is

\[ \square x. (Weight >= MaxValue \rightarrow (\diamond y. (PL1 \land y - x \geq 1s))). \]

In this paper, we use shorthand notations to represent the TPTL formulae. For the first two TPTL formulae, the notation is \( p \rightarrow q \ [t_{\min}, \ t_{\max}] \). Similarly, the notation \( p \ U \ q \ [t_{\min}, \ t_{\max}] \) corresponds to the last two formulae.

Besides the four TPTL property types above, we also leverage Perfume to derive other types of invariants as presented in Table I.

Type 1. The first invariant represents a variable that is assigned a certain value and kept for the duration of \( t_{\max} \). The second and the third invariants express that under the predicate \( expr \), the output holds for the duration of \( t_{\max} \).
Table I

<table>
<thead>
<tr>
<th>Type</th>
<th>Data-Time Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>var = [] [ ] t_{max} [ ] expr [ ] var = [] [ ] t_{max} [ ] expr U var = [] [ ] t_{max}</td>
</tr>
<tr>
<td>2</td>
<td>var = [] [ ] t_{min}, t_{max}</td>
</tr>
<tr>
<td>3</td>
<td>var = [] [ ] t_{min}, t_{max} [ ] var [ ] [ ] U [ ] [ ] var [ ] = [] [ ] [ ] t_{min}, t_{max}</td>
</tr>
<tr>
<td>4</td>
<td>expr [ ] [ ] [ ] var [ ] = [] [ ] [ ] t_{min}, t_{max} [ ] expr U var = [] [ ] [ ] t_{min}, t_{max}</td>
</tr>
</tbody>
</table>

**Type 2.** This invariant denotes the time interval that a variable takes a specific value.

**Type 3.** The two invariants describe a variable that is assigned different values with the time bounded by $t_{min}$ and $t_{max}$.

**Type 4.** These two invariants express the relationship between the condition $expr$ and the corresponding variable assignments in the time difference $t_{min}$ and $t_{max}$.

**D. Generate Data-Time Invariants**

The invariants dynamically detected by Daikon provide the relations of values of variables. After data invariants are generated, we process the trace file produced by Daikon’s front-end and apply Perfume to derive data-time invariants.

**Trace Process.** For each data invariant, we first extract all related data-trace records from the trace file. Each data-trace record includes runtime value information in one scan cycle. For the data invariant “$Weight \geq MaxValue$” $\implies PL1$ one of $\{false, true\}$ in the motivation example, we only extract the data-trace records corresponding to the predicate $Weight > MaxValue$ and the variable $PL1$ from the trace to detect the time boundary of the data invariant.

Given that the trace only contains the extracted data-trace records, we convert each record to a tuple $p_k = [(\text{name1}=\text{value1}, \text{name2}=\text{value2}, ..., \text{timestamp})]$, where $\text{name1}=\text{value1}, \text{name2}=\text{value2}, ...,$ is the variables’ names and values in predicate $expr$, $\text{name}=\text{value}$ is the name and value of the output variable $var$. For convenience, we denote tuple $p_k$ as $p_k = [\text{expr}, \text{var}]$.

A tuple $p_k$ contains the values of the variables contained in both $expr$ and $var$ in one scan cycle. The trace $\tau$ comprised of tuples $p_1, p_2, ..., p_k, ...$, represents all possible values that the variables in both $expr$ and $var$ obtain in one execution. $expr$ and $var$ will be evaluated true when particular combinations of input conditions are met.

**Deriving Invariants.** We propose four methods to derive different types of data-time invariants. If the predicate $expr$ evaluates true, we replace $name1=value1, name2=value2, ...,$ with the predicate. For instance, the tuple $[(\text{Weight}=27, \text{MaxValue}=26, \text{timestamp})]$ is replaced with $[(\text{Weight} \geq \text{MaxValue}, \text{timestamp})]$.

- **Method 1:** We extract sub-trace $s_n$ which is comprised of the tuples $p_1, p_2, ..., p_k$ that the output variable $var$ keeps a certain value and the predicate $expr$ evaluates true. We pass all of the sub-traces $s_1, s_2, ..., s_n$ to Perfume to obtain the maximum time that the causal relation $expr \rightarrow var$ holds.

- **Method 2:** We extract sub-trace $s_n$ which is comprised of the tuples $p_1, p_2, ..., p_k$ that the output variable $var$ keeps a certain value and the predicate $expr$ evaluates true. For the two consecutive sub-traces $(s_1, s_2)$, we keep the last tuple $p_k$ in $s_1$ and the first tuple $p_1$ in $s_2$. Similarly, for the two consecutive sub-traces $(s_2, s_3)$, we keep the last tuple $p_k$ in $s_2$ and the first tuple $p_1$ in $s_3$. The remaining sub-traces are processed so on and so forth. For the pairs of tuples $(s_1, p_k, s_2, p_1), (s_2, p_k, s_3, p_1), ...$, we pass them to Perfume and obtain the time interval that the causal relation $expr \rightarrow var$ holds.

- **Method 3:** We process the whole trace and only keep the tuples that represent state transitions. Formally, for the two consecutive tuples $p_{i-1}=[(\text{expr}_{i-1}, \text{var}_{i-1})]$ and $p_i=[(\text{expr}_i, \text{var}_i)]$, if one of the values changes in $p_i$ compared to $p_{i-1}$, then tuple $p_i$ keeps; otherwise, it will be removed. After the process finishes, we extract sub-trace $s_n$ which is comprised of the tuples $p=[(\text{expr}, \text{var})]$ and $p’=[(\text{expr’}, \text{var’})]$ that the expr evaluates true and $var’$ represents the occurrence of output event. In addition, $expr’$ does not necessarily evaluate true and $var$ does not necessarily represent the occurrence of output event. We pass all of the sub-traces $s_1, s_2, ..., s_n$ to Perfume to mine data-time invariants. The time boundary mined by this method corresponds to the time of state transition.

- **Method 4:** We extract the sub-trace $(s_n, t_n)$ which is comprised of the tuples $p=[(\text{expr}, \text{var})]$ and $p_i=[(\text{expr’}, \text{var’})]$ that the $expr$ evaluates true and $var’$ represents the occurrence of output event. In addition, $expr’$ does not necessarily evaluate true and $var$ does not necessarily represent the occurrence of output event. We exclude all $var$ from sub-trace $s_n$ and exclude all $expr’$ from sub-trace $t_n$.

1) The minimum time boundary $t_{min}$ between $expr$ and $var$: for sub-trace $(s_n, t_n)$, we only keep the first tuple in $s_n$. The remained tuples in $(s_n, t_n)$ are $(s_n, p_1, t_2, p_2, ..., t_n, p_k)$. We pass all of the sub-traces $(s_1, t_1), (s_2, t_2), ..., (s_n, t_n)$ to Perfume to mine data-time invariants.

2) The maximum time boundary $t_{max}$ between $expr$ and $var$: for sub-trace $(s_n, t_n)$, we only keep the first tuple in $t_n$. The remained tuples in $(s_n, t_n)$ are $(s_n, p_1, t_2, p_2, ..., t_n, p_k)$. We pass all of the sub-traces $(s_1, t_1), (s_2, t_2), ..., (s_n, t_n)$ to Perfume to mine data-time invariants.

In particular, there exists a case where the data invariant only includes one variable. In this case, the tuple $p_k = [(\text{name}=\text{value}, \text{timestamp})]$, and the above methods also hold. The trace process is simpler with only one variable involved. Take the data invariant GreenNS one of $\{false, true\}$ for example, we use Method 1 to derive the first Type 1 invariant and use Method 2 to derive the Type 2 invariant.

**E. Specification Mismatch**

The data-time invariants are more useful for describing the runtime behavior of the PLC program because they capture the
time constraints of data invariants. The dynamically detected invariants describe the behavior of the system, while the specifications express the expected behavior of a system. After deriving the data-time invariants, we check the dynamically detected invariants for errors against manually crafted specifications.

To examine whether there exists a mismatch between the observed and expected behavior of a program, for every generated data-time invariant \( \phi_i \) corresponding to the specifications, we manually check \( \sigma_i \models \phi_i \), where \( \sigma_i \in \Sigma \) is the actual specifications of PLC programs.

\section*{F. Test Case Generation}

To generate test cases, we follow the approach of SYsPLC [7]. The test suites generated by symbolic execution guarantee the instruction coverage. However, dynamic invariant detection requires adequate test cases to improve the quality of detected invariants. In the motivation example, if the automatically generated test case for the variable \( \text{Weight} \) is 27, Daikon will infer data invariant \( \text{Weight} == 27 \). If there are various values assign to the variable \( \text{Weight} \) like 47, -27, 87, 21, 28, 17, 23, 57, etc., the detected data invariant will not include \( \text{Weight} == 27 \), which is too concrete and makes no sense.

In this paper, we mutate existing test suites to make it suitable for dynamic invariant detection. The mutation operators are shown in Table II. Once the mutated test suites are generated, we remove the test cases which do not conform to the variable’s type and allowed value ranges. In addition, redundant test cases are discarded from the mutated test suites.

\section*{IV. Evaluation}

We apply our testing approach to a representative real-life case study, i.e., cosmetic packing. The production process is common to find in the automation industry. A code error has been found in the program of cosmetic packing. The case study is conducted on the Ubuntu 18.04 LTS operating system.

\subsection*{A. Experimental Setup}

The mateic project provides the iec2c compiler which generates ANSI C code equivalent to the original PLC program. Multiple C files are generated after code translation. To produce basic test suites, we employ KLEE [16] to perform symbolic execution in the translated C code. We use Python to create mutated test suites. During the instrument step, we employ ANTLR [17] to perform static analysis of the PLC program and then instrument translated ANSI C code with the result of static analysis. We use the gcc compiler to compile them into an executable softplc file. The executable softplc can simulate the run of the PLC program without running on a real PLC. We write a Python script to process trace files generated by Daikon’s front-end and apply Perfume to the processed trace file to derive data-time invariants. The source code of our implementation is available online [18].

\subsection*{B. Cosmetic Packing}

The cosmetic packing system packs cosmetics in a box and then puts the packed box into a hopper. Figure 3 shows the whole operation process. There are two conveyor belts C1 and C2. C1 sends cosmetics to the packing region and C2 sends the packed box to a hopper. The cosmetics move on C1 at a constant speed. Once a cosmetic reaches the packaging area, it will be put into a box. If there are three cosmetics in a box, the box will be packed one second later and C2 will transmit the box to a hopper. There are three switches Start, Pause, and Reset, which start, pause, and reset the process, respectively.

The specifications of the automated packing system are: I. The time interval for two consecutive cosmetics to reach the packaging area is 1.5 seconds since the packing process consumes time. II. The packed box should be present in C2 within one second after the third cosmetic is put into the box.

We execute the code in the simulator for 3000 cycles and repeats five times. Each scan cycle lasts for 50 milliseconds. There are total of 7 data-time invariants obtained, and the invariants corresponding to the specifications are listed in Table III. In Table III, the first invariant associated with Specification I represents that the two consecutive cosmetics reaches the packaging area for at least 1.5 seconds. The second invariant associated with Specification I denotes that the minimum time duration that cosmetics arrive in the packaging area is 1.5 seconds. These two invariants satisfy the specification.

An implementation error has been detected by applying our approach. In the process of cosmetic packing, it takes one second to pack the box after the third cosmetic reaches the packaging area. However, for the data-time invariants derived in one simulation, the time of box packing lasts for nearly two seconds. As shown in Specification II of Table III, the

\begin{table}[h]
\centering
\caption{Mutation Operators Used in Mutating Test Cases}
\label{table:mutation_operators}
\begin{tabular}{ll}
\hline
Mutation Operator & Example \\
\hline
boolean constant replacement & \( \text{var=true} \rightarrow \text{var=false} \) \\
numeric constant replacement & \( \text{var = 1.0} \rightarrow \text{var = 2.0} \) \\
unary operator insertion & \( \text{var=5} \rightarrow \text{var=-5} \) \\
\hline
\end{tabular}
\end{table}

\begin{table}[h]
\centering
\caption{Detected Invariants of Cosmetic Packing}
\label{table:detected_invariants}
\begin{tabular}{l|l}
\hline
Specification & Data-Time Invariant \\
\hline
I & reached=true \[1.5s, 3s\] \\
& reached=false U reached=true \[1.5s, 1.5s\] \\
\hline
II & full_box \rightarrow send_box \[1.57s, 1.57s\] \\
\hline
\end{tabular}
\end{table}
detected packing time violates the specification. Issues might occur since cosmetics repeatedly approach the packing area every 1.5 seconds.

We analyze the implemented program and simulator to find the cause of specification violations. The reason is that the timer of box packing returns to zero when we pause the packing process after the system operates for 6.5 seconds. After restarting the system, the timer starts to time from zero, which is inaccurate since the box packing process can keep its operation after restarting the system. We update the implementation and the detected invariant \( \text{full\_box} \rightarrow \text{send\_box} \{1s, 1s\} \) satisfies the specification. Concretely, in the original implementation, we use an \( \text{on-delay timer} \) (TON) to time the box packing. However, TON does not retain the elapsed time if the input goes false. We switch TON to \( \text{retentive on-delay timer} \) (RTO) which retains the elapsed time when the pause switch is pressed.

C. Discussion

The scalability problem arises when the number of program points instrumented increases during performing dynamic invariant detection. In addition, the number of scan cycles executed also brings time and space costs. To improve scalability in the automatic testing process, one could reduce either the number of program points instrumented or the number of scan cycles. Reducing the number of cycles executed in one execution and repeating several times execution can tackle the scalability issue.

V. RELATED WORK

Sallai et al. [19] generate x86-representations of PLC programs to test, simulate, and visualize PLC programs. They transform PLC programs into C, Scilab, and Java programs. The semantically equivalent x86 representation which can execute on personal computers overcomes the lack of advanced tools to help PLC programming. Our work converts PLC programs into C to simulate the execution of the PLC code.

PLCInspector [20] mines either linear temporal logic (LTL) specification using Texada or data invariants using Daikon from runtime traces of PLC programs. PLCInspector does not combine Texada with Daikon to detect data-temporal properties. Besides, the mined LTL specification cannot describe the runtime behavior of PLC programs properly. As PLC runs in real-time, LTL is not suitable to quantitatively express the time boundary of the event occurring.

The work most related to ours is \( V_{\text{ET}} \)PLC [12], which infers events from traces based on value changes then uses Perfume to mine temporal invariants to uncover time intervals between different events. \( V_{\text{ET}} \)PLC uses mined invariants to generate timed event sequences that serve as inputs for automated safety vetting PLC code. Our approach uses Perfume to find time bounds for the data invariants from the execution traces.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a novel approach to test PLC programs by mining invariants from execution traces. Our approach dynamically detects combined data-time invariants to observe the time performance of implementation programs. The derived invariants expressing the runtime behavior of the PLC programs are checked with the specifications. We evaluate our approach in a representative real-life scenario.

The evaluation results show that our approach is useful for discovering implementation errors that are difficult to find using existing methods.

The dynamic detection technique proposed in this paper to derive data-time invariants is dedicated to the test of PLC programs. Since timing constraints are essential in many embedded systems or cyber-physical systems, we plan to apply this paper’s method to test a broader class of systems beyond the scope of industrial control systems as one of the future works.

Acknowledgment. This work is partially supported by NKRDP (2019YFB2102602).

REFERENCES


Abstract—Constraint solving is a key challenge in symbolic execution. Usually, symbolic execution uses the fixed-size bit-vector theory to precisely model the program’s behavior and generates the bit-vector formula to query the SMT solver. To solve such bit-vector formula, SMT solvers usually adopt a bit-blasting and conjunctive normal form (CNF) conversion step, transforming the original formula into an equi-satisfiable CNF formula, and then check the formula’s satisfiability. However, the different CNF conversions can significantly affect the efficiency of SAT solving. We observe that each CNF encoding algorithm has its suitable applications, while adopting a specific CNF conversion algorithm for all formulas is often not optimal. Therefore, we propose to intelligently select a suitable CNF encoding algorithm for each logical formula. We have integrated our selection algorithm into the symbolic execution framework based on KLEE and STP, which are the state-of-the-art symbolic execution engine for C programs and its default underlying constraint solver, respectively. The experimental results, based on extensive evaluation of 86 real-world C programs in Coreutils benchmark, indicate that our method can effectively improve the efficiency of symbolic execution. On average, our method increases the number of the explored paths by 27.2%.

Index Terms—CNF, SAT, Machine learning, Symbolic execution

I. INTRODUCTION

Symbolic execution [8], [11] is a widely used program analysis technique to systematically explore the path space of programs. Its applications covers many fields of software engineering, including automated test generation, software verification and bug detection. Symbolic execution is processing on symbolic inputs instead of concrete inputs. Therefore, the operations in program are recorded as the computation between symbolic expressions. For each program path, symbolic execution maintains a path condition (PC) that is updated whenever a branch instruction is encountered. Only if current branch is reachable is the corresponding path condition updated. Otherwise, the branch is unreachable and symbolic execution terminates the exploration. Note that the feasibility of a program path is determined by the result of constraint solving, i.e., determining whether the path condition which is a quantifier-free first-order logic formula [13] is satisfiable. In this way, symbolic execution can explore the path space of the program systematically and understand the program precisely. Due to these advantages, many successful symbolic execution engines emerge, such as KLEE [3], Pex [23], and SPF [18], to name a few.

Obviously, constraint solving is a critical component of symbolic execution, as it is used to check the feasibility of a path and generate the test to execute the corresponding path if feasible. However, there exists many obstacles for constraint solving, which further limits the development of symbolic execution. On the one hand, the number of paths to be explored grows exponentially with the increase of program size and some syntactic constructs like loops can even lead to infinite paths. Therefore, symbolic execution engines will issue a bulk of queries to the underlying solver for complex programs. On the other hand, because of complex features in real world programs, i.e., non-linear arithmetic and array operation, symbolic execution engines will build complex queries which are quite hard to solve. In brief, constraint solving is the most time-consuming part and limits the scalability of symbolic execution.

In general, symbolic execution uses bit-vector arithmetic SMT theory combining with other SMT theories (e.g., array theory) to precisely model the behavior of program. When solving the bit-vector formula, bit-blasting is a key step in most SMT solvers which reduces a bit-vector formula into a pure propositional SAT formula. Unfortunately, such SAT formula won’t be solved by SAT solvers immediately. Modern SAT solvers [7] mainly take the input as a conjunctive normal form (CNF) formula in which the solver is able to apply highly efficient solving algorithms. Consequently, SMT solvers have to convert the SAT formula after bit-blasting into an equi-satisfiable CNF formula, which can be efficiently solved by
SAT solver.

Currently, there are two common CNF transformation algorithms: Tseitin algorithm [25] and the algorithm based on technology mapping [6]. The former adds a new variable to each logic gate of the original formula, and then constrains the variable with a new clause to form a new CNF. This algorithm has a lower complexity, but the generated CNF is often huge and difficult to solve. The latter divides AIG (And Inverter Graph) into logical nodes wherein there is no more than K inputs for each node, and extracts CNF for each node based on a look-up table. The algorithm has a higher complexity, but it can generate CNF that is more concise and easier to solve. We have the following key observation of CNF conversion in SMT solvers: almost every SMT solver of QF_BV logic always uses one of the specific CNF conversion algorithms above. However, each CNF conversion algorithm has its suitable applications, and the efficiency of using a specific CNF conversion algorithm as the solution of all formulas is often not optimal.

If the propositional formulas can be classified according to their suitable CNF conversion algorithm, then the solving efficiency of SMT solver can be improved distinctly. Therefore, an intuitive idea is to extract the features of propositional formulas, and divide propositional formulas into two different categories, the one with higher efficiency of SMT solving using Tseitin algorithm and the one using Technology mapping algorithm. Then using machine learning to train a model which well classify such two different categories. Machine learning is a branch of artificial intelligence that focuses on building models that learns from data and improve their accuracy over time. In machine learning, models are trained to find the law of large amounts of training data, so that the model’s predictions on new data can be made as correct as possible. Machine learning algorithms can be divided into supervised learning, unsupervised learning and reinforcement learning according to learning methods. In real world, the application of machine learning is very extensive, such as: data mining and analysis, pattern recognition and many other fields. As for SMT solving, machine learning also has many combined applications [1], [21].

This paper proposes to select a suitable CNF encoding algorithm for each given formulas. Our key idea is to use the existing SMT formulas in the SMT-LIB benchmark repository [2] as training data to train a machine learning model offline, so as to automatically choose a more appropriate CNF encoding algorithm for the formula in the process of SMT solving, hoping to improve the efficiency of SMT solving. We have implemented our approach on KLEE and STP, which are the state-of-the-art symbolic execution engine for C programs and its default underlying constraint solver, respectively. The experimental results, based on extensive evaluation of 86 real-world C programs in Coreutils benchmark, indicate that our method can effectively improve the efficiency of symbolic execution. On average, our method increases the number of the explored paths by 27.2%

The remainder of the paper is organized as follow. Section 2 shows the related work. Section 3 illustrates our method in details. Section 4 gives the evaluation and Section 5 discuss the limitation. Finally we draw a conclusion of the paper.

II. RELATED WORK

Our work is closely related to the constraint solving optimization in symbolic execution and machine learning techniques in constraint solving. We will discuss them in detail.

A. Constraint Solving Optimization in Symbolic Execution

The ability of constraint solving is the main bottleneck for the scalability of symbolic execution. Therefore, lots of research focus on accelerating constraint solving in symbolic execution. A typical idea is optimizing constraint solving in the context of symbolic execution, which mainly focuses on the optimizations of symbolic expression and invokes the underlying solver in a black-box manner [3], [10]. CUTE [22] has implemented a mechanism of fast unsatisfiability check based on the syntactical contradiction of symbolic expression, which reduces invocations of constraint solver by 60-95%. KLEE [3] uses three kinds of optimizations to speed up constraint solving, including caching the counter-examples to avoid calling the underlying solver in certain situations, rewriting the constraint into simpler one, e.g. strength reduction and linear simplifications, just like what a compiler does, and splitting the constraint into disjoint sets of independent constraints for better reusing. Aiming at array constraint, KLEE-Array [20] proposes some novel optimizations based on repeated values in constant arrays to simplify the symbolic expressions. In addition, there exists some works which synthesize symbolic execution and constraint solving and then use the constraint solver in a white-box manner. For example, multiplex symbolic execution (MuSE) [31] collects all partial solutions generated by the underlying constraint solver in one time of solving and constructs multiple program inputs according to these solutions.

B. Machine Learning Techniques in Constraint Solving

Recently, machine learning is a hot topic in academia and industry, with new methods invented all the time. Researchers in different research areas benefit from emerging machine learning techniques a lot. In constraint solving, some researchers try to improve the ability of constraint solver by combining machine learning techniques. Portfolio-based approach is an well-known way to improve the efficiency of constraint solver with machine learning methods, such as SATZilla [27], CPHydra [17] and MachSMT [21]. The basic idea is picking a solving algorithm from a set of solving algorithms, which is a typical classifier problem that machine learning method is good at. MLB [15] transforms the feasibility problem of the path condition in symbolic execution into optimization problem and employs an optimization solver which implements a machine learning guided sampling and validation method. FastSMT [1] is designed to generate a faster solving strategy for SMT solving. First, it uses a combining method of random search and neural
network to learn a set of candidate solving strategies. Then it synthesizes a combined solving strategy with branches based on the candidates. Besides, Petr Somol et al. proposed a search principle for optimal feature subset selection using the Branch & Bound method [26], which can be used to improve performance of SAT solvers. Earlier research [14] accelerated the SMT solving by learning to select branching rules in DPLL algorithm.

III. THE PROPOSED APPROACH

This section presents the details of our intelligent selection method. The framework will be introduced first. Then, the extraction of formula features and the CNF encoding selection are explained in the next two sub-sections.

A. Framework

Algorithm 1 shows details of our intelligent selection method of CNF encoding. The inputs are a logical formula \( \text{formula} \) represented in the SMT-LIB format [2]. The algorithm first employs AST to translate input \( \text{formula} \) to Abstract Syntax Tree representation, \( T \) (Line 1). Then, we apply \( \text{MERGE} \) to merge the leaf nodes of \( T \) (Line 2) which represent same variables or constants. \( \text{MERGE} \) returns a directed acyclic graph (DAG) \( D \). Next, the algorithm carries out \( \text{EXTRACT} \) (c.f. Algorithm 2) on \( D \). \( \text{EXTRACT} \) returns the corresponding feature \( F \). Finally, we use an intelligent selection method on \( F \) to select the most effective CNF encoding for given logical formula.

Algorithm 1 ISCE(\( \text{formula} \))

Input: The SMT formula \( \text{formula} \).
Output: The CNF encoding method Result.
1: \( T = \text{AST}(\text{formula}) \)
2: \( D = \text{MERGE}(T) \)
3: \( F = \text{EXTRACT}(D) \)
4: \( \text{Result} = \text{SELECT}(F) \)
5: return \( \text{Result} \)

B. Feature Extraction

Algorithm 2 gives the details of feature extraction from the original formula. The input is a DAG which represents a logic formula compactly, the output is the representation in bag of word model [30].

Specifically, The algorithm considers nodes in DAG as words, and uses the type of nodes to distinguish them, and count the number of nodes in different types. Consider the following example,

\[
x_1 \rightarrow (x_2 \land x_3)
\]

There are three kinds of node types, \( i.e. \) variable, \( \rightarrow \) and \( \land \). The corresponding BoW representation is,

\[
\{ \text{variable} : 3, \rightarrow : 1, \land : 1 \}
\]

which keys are node types and values are the number of nodes in different types.

\begin{algorithm}
\caption{EXTRACT(D)}
\textbf{Input:} The DAG of formula \( D \).
\textbf{Output:} The BoW representation \( \text{BoW} \).
1: \( N = \text{NODES}(D) \)
2: \textbf{for} node \( \in N \) \textbf{do}
3: \( \text{BoW}[\text{node}] \leftarrow \text{BoW}[\text{node}] + 1 \)
4: \textbf{end for}
5: return \( \text{BoW} \)
\end{algorithm}

C. Intelligent Selection

The Algorithm 1 uses \( \text{SELECT} \) to get the most suitable CNF encoding algorithm, which improves the solving efficiency of SMT solver apparently. The input is feature of a logical formula, which is generated by III-B. The output is Tseitin algorithm or Technology mapping algorithm which improves the solving efficiency most of SMT solver.

We employ an offline trained learning model to predict a CNF encoding algorithm for an logical formula. To train the model, we generate the training data from the existing SMT formula in the SMT-LIB benchmark repository [2]. Each element in the training data is a tuple \( (\mathcal{E}(\varphi), t) \) consisting of four parts: \( \mathcal{E}(\varphi) \) is the embedding feature of the current formula \( \varphi \), \( t \) is the specific CNF encoding algorithm which improves the speed of SMT solving more than other \( (c.f. t = 0 \) means Technology mapping is better and \( t = 1 \) means Tseitin algorithm). Since we are interesting in analyzing computer programs, we choose the formulas in QF BV and QF ABV logic, and generate the corresponding embedding feature by III-B. For \( t \) of each element in the training data, we use STP [7] as SMT solver under Technology mapping and Tseitin algorithm simultaneously, then set \( t \) to the algorithm that spending less time when solve formula \( \varphi \).

D. Symbolic Execution Framework

This sub-section depicts how our intelligent selection method can be integrated into the symbolic execution framework. Algorithm 3 gives the symbolic execution framework. The input is the program under symbolic execution. Our framework adopts a state-based symbolic execution [11] and employs a worklist based implementation. In the beginning, there is only initial state \( s_i \) in the worklist \( (c.f. \text{Line 1}) \).

The main loop is a worklist based procedure. When exploring the state space, the symbolic executor selects a state from the worklist to explore the state space (Line 5). During symbolic execution, logical formula of corresponding path condition is generated (Line 6). Then we use our intelligent selection method to decide which CNF encoding algorithm should be used so that SMT solver may be speed up (Line 8). Finally, the CNF encoding algorithm \( En_{\varphi} \) is applying to speedup the SMT solver and the symbolic executor would append new states into worklist (Line 18).

The intelligent selection needs to balance the effectiveness and selection overhead. In principle, we can have a trained learning model that can recommend the best CNF encoding algorithm for each logical formula in validation set. However,
the selection introduces more overhead which consist of feature extraction and learning model prediction. This balance is controlled by a variable $K$. A variable $T$ is initialize to 0. We use $T$ to count the times our method continuously predicts the same CNF encoding algorithm. We use $Save_{en}$ to save the previous prediction. When $T$ grows to $K$, we no longer use our selection algorithm but use the $Save_{en}$ to reduce overhead. In our experiments, we set $K$ to 100.

IV. EXPERIMENTS

We have implemented our method on KLEE [3] (i.e. a state-of-the-art engine for C programs). KLEE’s version is 2.3-pre. We use STP as the backend solver and bit-vector SMT theory for encoding the path constraints. STP’s version is 2.3.3. We train the intelligent selection model by XGBoost [4]. We use the AST translation and Bag of Word embedding based on jSMTLIB [5].

We have conducted extensive experiments to answer the following two research questions:

- **RQ1**: what is the performance impact of the XGBoost intelligent selection algorithm?
- **RQ2**: how effective is our intelligent selection algorithm?

Here, effectiveness means exploring more paths during symbolic execution.

A. Experimental Setup

To evaluate the effectiveness of our method, we use Coreutils as the benchmark. Coreutils is the mainstream benchmark for the symbolic execution researches whose implementations are based on KLEE. The used Coreutils’s version is 6.11. There are 89 programs (46746 SLOCs) in total.

We train the XGBoost model for intelligent selection as follows. We use the QF_BV, QF_ABV SMT-LIB2 benchmarks [2] for generating the data set. We filter the formulas whose ASTs contain more than 50,000 nodes. We use the bag of words (BoW) model [30] and the one-hot encoding [9] as the embedding feature of the logical formulas and the CNF encoding algorithm, respectively. We use STP [7] under Tseitin and Technology mapping algorithm to find the most suitable CNF encoding for every formula in our benchmarks.

The timeout threshold is set to 30 seconds. If timeout occurred both Tseitin and Technology mapping algorithm, we would filter the corresponding formula.

We compare our method (which implements based on XGBoost) with the one employing Multi-layer Perceptron classifier from sklearn [19], to show what is the performance impact of the XGBoost algorithm. We have 18,782 formulas after filtered in above way. We select 50% for training dataset and the others for validation sets. XGBoost uses default settings. For MLP in sklearn, we use adam as solver, the hidden layer sizes is $(30, 60, 30, 10)$ and the activation function is logistic.

We compare our symbolic execution framework with intelligent selection integrating, with baseline KLEE under two search heuristics, i.e., DFS and BFS. We analyze each Coreutils program in 30 minutes. We set the end condition of intelligent selection (c.f. Algorithm 3 Line 8) as intelligent selection generating same continuous results more then $K$ times. $K$ is a threshold that we set it to 100 in our experiments. We used the same options as KLEE mentions in [3]. But we close three optimizations, i.e., constraint independence, counterexample cache and branch cache, to generate more queries to smt solver.

All the experiments were carried out on a Server with 64GB memory and 16 3.1 GHz cores. The operating system is Ubuntu 14.04.

B. Experimental Results

**Answer to RQ1.** To answer the first question, we evaluate our XGBoost based intelligent selection by comparing with MLP (Multi-layer Perceptron classifier) classifier based version in three aspects: accuracy, recall and confusion matrix [24].

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>XGBoost</td>
<td>91%</td>
<td>89%</td>
</tr>
<tr>
<td>MLP</td>
<td>91%</td>
<td>83%</td>
</tr>
</tbody>
</table>

Table I shows the accuracy and recall of different machine learning model. XGBoost has the same accuracy as MLP but higher recall. Our dataset consists of 3,025 formulas that is suitable for Technology mapping algorithm and 15,757 formulas for Tseitin algorithm. As our data is imbalance, where there are different number of samples in each class, the recall is more important than accuracy.

Table II and III are confusion matrix of XGBoost and MLP, respectively. The column names and row names, *Map* or

---

**Algorithm 3 SE($P$)**

**Input:** A program $P$.  
1: `worklist = \{s_i\}` 
2: $T = 0$ 
3: $Save_{en} = \text{default}$ 
4: while `worklist ≠ ∅` do 
5: $s = \text{Choose(worklist)}$ 
6: $C = \text{GenConstraints}(P, s)$ 
7: if $T < K$ then 
8: $En_{cnf} = \text{ISCE}(C)$ 
9: if $En_{cnf}$ is $Save_{en}$ then 
10: $T = T + 1$ 
11: else 
12: $T = 1$ 
13: $Save_{en} = En_{cnf}$ 
14: end if 
15: else 
16: $En_{cnf} = Save_{en}$ 
17: end if 
18: `worklist ← worklist ∪ Execute(s, En_{cnf})` 
19: end while
Tseitin, means the number of formulas that solved efficiently when encoding to CNF by Technology mapping or Tseitin algorithm. In Table II, of 3,000 formulas classified to $Map$ (c.f., first line), XGBoost judged that 2,618 were $Map$. But In Table III, MLP judged 2,095 were $Map$ of the same 3,000 formulas. XGBoost predicts 523 samples correctly more than MLP, which is 17% in $Map$ class.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Map$</td>
<td>2618</td>
</tr>
<tr>
<td>$Tseitin$</td>
<td>1357</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Map$</td>
<td>2095</td>
</tr>
<tr>
<td>$Tseitin$</td>
<td>670</td>
</tr>
</tbody>
</table>

**Answer to RQ1:** XGBoost have better performance than MLP on recall in the imbalance dataset. More specifically, XGBoost correctly predicts 17% of the samples on the minority class.

**Answer to RQ2.** To answer the second research question, we compare our symbolic execution framework with intelligent selection integrating, with baseline KLEE. We evaluate them in path number which have been explored during symbolic execution.

Figure 1&2 show the comparison results of new paths in BFS and BFS, respectively. The X-axis shows the benchmark programs ordered by the values in Y-axis. The Y-axis shows the relative increasing of the explored paths, which is defined as follows, where $N_{OPT}$ denote the number of paths explored after employing our method, and $N_{BASELINE}$ represents the number of original symbolic execution.

$$\frac{N_{OPT} - N_{BASELINE}}{N_{BASELINE}}$$  (3)

As shown by Figure 1, our method can improves the number of explored paths on 62(73%) programs. On the other hand, there are 24(27%) programs on which we decrease the number of paths because of the feature extraction overhead; however, the decreasing is slight, i.e., -2.81% (-5.2%~ -0.14%) on average. Our method can on average improves the number of explored paths by 26.7% (-36%~522%).

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Map$</td>
<td>2618</td>
</tr>
<tr>
<td>$Tseitin$</td>
<td>1357</td>
</tr>
</tbody>
</table>

**Answer to RQ2:** Our method is effective to improve symbolic execution’s ability of path exploration. On average, our method increases the number of paths by 27.2%.

**V. THREAT TO VALIDITY**

The external validity is a major threat to our experimental results. It is mainly due to the limited benchmark we used and the generalization of machine learning model. For the former, although the number and type of benchmark may be insufficient, Coreutils is a widely used benchmark for evaluating the performance of symbolic execution [3], [16], [28], and the current experimental results have demonstrated the effectiveness of our method. However, we plan to evaluate our prototypes on more benchmarks in the next step.

**VI. CONCLUSION**

In this paper, we propose a method to intelligently select a suitable CNF encoding algorithm for a given logical formula, which is more efficient for constraint solving than the one using a specific CNF encoding algorithm for all formulas. Our
approach leverages offline trained machine learning models to
predict the suitable CNF encoding algorithm for a given logical
formula. We integrate our selection algorithm into the symbolic execution framework based on KLEE and STP, which are
the state-of-the-art symbolic execution engine for C programs
and its default underlying constraint solver, respectively. The
experimental results, based on extensive evaluation of 86 realworld C programs in Coreutils benchmark, indicate that our
method can effectively improve the efficiency of symbolic
execution. On average, our method increases the number of
the explored paths by 27.2%.
ACKNOWLEDGMENT
This work was supported by National Natural Science
Foundation of China (No.61632015)
R EFERENCES
formulas,” in Advances in Neural Information Processing Systems 31:
Annual Conference on Neural Information Processing Systems 2018,
NeurIPS 2018, December 3-8, 2018, Montréal, Canada, S. Bengio,
H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and
in 8th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2008, December 8-10, 2008, San Diego, California, USA,
Proceedings, R. Draves and R. van Renesse, Eds. USENIX Association,
in Proceedings of the 22nd ACM SIGKDD International Conference
on Knowledge Discovery and Data Mining, San Francisco, CA, USA,
August 13-17, 2016, B. Krishnapuram, M. Shah, A. J. Smola, C. C.
in NASA Formal Methods - Third International Symposium, NFM 2011,
Pasadena, CA, USA, April 18-20, 2011. Proceedings, ser. Lecture Notes
in Computer Science, M. G. Bobaru, K. Havelund, G. J. Holzmann, and
speeding up SAT,” in Theory and Applications of Satisfiability Testing SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31,
arrays,” in Computer Aided Verification, 19th International Conference,
random testing,” in Proceedings of the ACM SIGPLAN 2005 Conference
on Programming Language Design and Implementation, Chicago, IL,
USA, June 12-15, 2005, V. Sarkar and M. W. Hall, Eds. ACM, 2005,
pp. 213–223.
International Symposium on Software Testing and Analysis, ISSTA 2015,
Baltimore, MD, USA, July 12-17, 2015, M. Young and T. Xie, Eds.
a review of classification and combining techniques,” Artif. Intell. Rev.,

[15] X. Li, Y. Liang, H. Qian, Y. Hu, L. Bu, Y. Yu, X. Chen, and X. Li,
“Symbolic execution of complex program driven by machine learning
based constraint solving,” in Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, ASE 2016,
[16] K. Ma, Y. P. Khoo, J. S. Foster, and M. Hicks, “Directed symbolic
execution,” in Static Analysis - 18th International Symposium, SAS 2011,
95–111.
“Using case-based reasoning in an algorithm portfolio for constraint
solving,” in Irish conference on artificial intelligence and cognitive
Conference on Automated Software Engineering, Antwerp, Belgium,
[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion,
constraints in symbolic execution,” in Proceedings of the 26th ACM
SIGSOFT International Symposium on Software Testing and Analysis,
Santa Barbara, CA, USA, July 10 - 14, 2017, T. Bultan and K. Sen,
solvers,” 2020.
for C,” in Proceedings of the 10th European Software Engineering
Conference held jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2005, Lisbon, Portugal,
September 5-9, 2005, M. Wermelinger and H. C. Gall, Eds. ACM,
[23] N. Tillmann and J. de Halleux, “Pex-white box test generation for .net,”
in Tests and Proofs - 2nd International Conference, TAP 2008, Prato,
Italy, April 9-11, 2008. Proceedings, ser. Lecture Notes in Computer
Science, B. Beckert and R. Hähnle, Eds., vol. 4966. Springer, 2008,
and their applications in model checking,” Proc. IEEE, vol. 103, no. 11,
concolic execution engine tailored for hybrid fuzzing,” in 27th USENIX
Security Symposium, USENIX Security 2018, Baltimore, MD, USA,
August 15-17, 2018, W. Enck and A. P. Felt, Eds. USENIX Association,
[31] Y. Zhang, Z. Chen, Z. Shuai, T. Zhang, K. Li, and J. Wang, “Multiplex
symbolic execution: Exploring multiple paths by solving once,” in 35th

543


Graph queries for analyzing the coverage of requirements by test cases

Shingo Ariwaka, Hiroyuki Nakagawa, Tatsuhiro Tsuchiya
Graduate School of Information Science and Technology, Osaka University

Abstract

We study the applicability of graph queries to the coverage analysis of test cases for requirements specifications. First we show that when the similarity degrees between requirements specifications and test cases are available, they can be represented in the form of a graph. Then we identify several queries that are useful for extracting coverage information and show that all these queries can be written in the Cypher query language, a common graph query language. In a case study we apply these queries to data obtained from a real-world project in industry. The results of the case study show that coverage information can be retrieved in reasonable time. We also compare the graph queries with SQL queries with respect to conciseness and processing time.

1. Introduction

In this paper we study the applicability of graph queries in the field of software engineering. A graph query is a query for accessing a graph database which manages data with a graph structure. Graph databases and graph queries are becoming popular since they are specially tailored to handle data with graph structures, such as social networks [1]. On the other hand, applications of graph databases and graph queries to software engineering problems have not been studied sufficiently. This paper aims to investigate whether or not graph queries can be effectively used in analyzing the test coverage of requirements specifications in software system development. In our context, coverage means how many of the requirements specifications are tested by the test cases and the extent to which each of the requirements specification is tested.

There has already been a body of research that can be used to automatically compute the relevance or similarity between software artifacts, such as test cases and requirements specifications (for example, [2, 3]). In these previous studies, the similarity degree between two artifacts is estimated using natural language processing. The obtained similarity degrees can be used to, for example, infer the existence of traceability link between them.

In this paper, we assume that using some of these techniques similarity degrees have already been computed between test cases and requirements specifications. Under the assumption, we first identify several information items that are useful for system developers to perform coverage analysis. Then we show that these items can be naturally specified in the form of graph queries which are in turn used to retrieve coverage information from a graph database. Furthermore we demonstrate practical applicability of these graph queries through a case study using data obtained from a real-world industrial product.

The structure of this paper is as follows. In Section 2, we introduce graph databases and Cypher, a well-known query language for graph databases. In Section 3, we describe the basic assumptions about requirements specifications and test cases and show how they can be represented in the form of a graph. In Section 4, we list information items that can be useful for developers to perform coverage analysis. We also show that these items can be retrieved using Cypher queries. In Section 5, we show the results of a case study where a data set from an industrial product is used. In Section 6 we describe related research. In Section 7 we discuss potential threats to validity of the study. Finally, in Section 8, we summarize the paper and discuss future research directions.

2. Graph database and query languages

2.1. Graph database

A graph database is a database that manages data with a graph structure. In this paper we adopt the model of graph structures used by Neo4j, a well-known graph database [1]. In the Neo4j’s model, a graph structure consists of nodes, relationships, and properties. Nodes refer to vertices and relationships refer to direct edges in graph theoretical terms. Properties are attributes assigned to nodes and relationships and can hold data such as numbers and strings in a key-value format. Furthermore, labels can be assigned to nodes and relationships. When searching the database, the labels are useful for specifying data items to be retrieved.

DOI 10.18293/SEKE2021-122
2.2. Cypher query language

In database management systems, accesses to databases are performed through execution of queries written in a query language. Cypher is a common query language for graph databases [1]. For example, Neo4j uses Cypher as its query language. Cypher allows for expressive querying of graph databases. Queries in Cypher create, read, update, and delete data items. In this paper we basically focus on read queries since our interest is in analysis over a set of requirements specifications and test cases that have already been provided. Read queries in Cypher start with keyword MATCH followed by a search pattern for finding nodes or relationships. Additional constraints to the pattern can be added using keyword WHERE.

Figure 1 shows a graph data representing relationships among three peoples which represents that Smith is known to Williams and Johnson. The nodes correspond to the people and represent their names by property name. For example, a query to retrieve a list of people who know Smith from this data can be written as follows.

![Figure 1: A graph data representing human relationships](image)

1 MATCH (a:Person)-[:Knows]->(b:Person)
2 WHERE b.name = "Smith"
3 RETURN a

Table 1: Example of traceability data consisting of seven requirements specifications and ten test cases. Entities with 0 similarity are omitted.

(a) Requirements specifications-pairs

<table>
<thead>
<tr>
<th>r0</th>
<th>r1</th>
<th>0.14</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td>r5</td>
<td>0.08</td>
</tr>
<tr>
<td>r0</td>
<td>r6</td>
<td>0.59</td>
</tr>
<tr>
<td>r1</td>
<td>r4</td>
<td>0.97</td>
</tr>
<tr>
<td>r1</td>
<td>r5</td>
<td>0.84</td>
</tr>
<tr>
<td>r2</td>
<td>r3</td>
<td>0.2</td>
</tr>
<tr>
<td>r2</td>
<td>r4</td>
<td>0.94</td>
</tr>
<tr>
<td>r2</td>
<td>r6</td>
<td>0.34</td>
</tr>
<tr>
<td>r4</td>
<td>r5</td>
<td>0.93</td>
</tr>
</tbody>
</table>

(b) Test case and requirements specification-pairs

<table>
<thead>
<tr>
<th>t0</th>
<th>r0</th>
<th>0.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>r0</td>
<td>0.57</td>
</tr>
<tr>
<td>t1</td>
<td>r1</td>
<td>0.82</td>
</tr>
<tr>
<td>t2</td>
<td>r2</td>
<td>0.35</td>
</tr>
<tr>
<td>t3</td>
<td>r2</td>
<td>0.16</td>
</tr>
<tr>
<td>t4</td>
<td>r2</td>
<td>0.59</td>
</tr>
<tr>
<td>t4</td>
<td>r4</td>
<td>0.21</td>
</tr>
<tr>
<td>t5</td>
<td>r4</td>
<td>0.01</td>
</tr>
<tr>
<td>t5</td>
<td>r5</td>
<td>0.03</td>
</tr>
<tr>
<td>t7</td>
<td>r1</td>
<td>0.91</td>
</tr>
<tr>
<td>t7</td>
<td>r2</td>
<td>0.41</td>
</tr>
<tr>
<td>t7</td>
<td>r4</td>
<td>0.09</td>
</tr>
<tr>
<td>t8</td>
<td>r0</td>
<td>0.06</td>
</tr>
<tr>
<td>t8</td>
<td>r1</td>
<td>0.28</td>
</tr>
<tr>
<td>t8</td>
<td>r2</td>
<td>0.48</td>
</tr>
<tr>
<td>t8</td>
<td>r5</td>
<td>0.5</td>
</tr>
<tr>
<td>t9</td>
<td>r1</td>
<td>0.21</td>
</tr>
<tr>
<td>t9</td>
<td>r2</td>
<td>0.66</td>
</tr>
<tr>
<td>t9</td>
<td>r5</td>
<td>0.54</td>
</tr>
</tbody>
</table>

1.2. Graph structure as traceability data

The traceability data can be represented in a graph structure as follows. The test cases and requirements specifications correspond to nodes. Nodes of test cases and nodes of requirements specifications are distinguished by assigning different labels to them. We call the node corresponding to a test case a test case node and the node corresponding to a requirements specification a requirement node. The similarity degree between requirements specifications pair or between test cases and requirements specifications is represented by a property assigned to the relationship defined between the requirement nodes or between the test case node and the requirement nodes. An exception is when the similarity degree between two nodes is 0, in which case no relationship is defined between them.
It should be noted that the directions of relationships are irrelevant to the traceability data we consider. Hence we set at most one relationship of either direction between two nodes. This is a standard treatment of undirected edges in the graph model of Neo4j.

Figure 2 shows a visualization of the above example, obtained using Neo4j’s functions.

**4. Database queries for coverage analysis**

In this section, we list some information items that may be useful for developers to know how well the test cases test the requirements specifications. We wrote queries both in Cypher and SQL for retrieval of all these items. For space limitations we select two items as examples and present Cypher queries and SQL queries for extracting them. In addition the conciseness of the queries is evaluated in terms of character count.

**4.1. Coverage information to be extracted**

We identify a total of seven information items that can be useful for coverage analysis as follows:

1. List of test cases that directly test requirements specification \( R \)
2. List of test cases that directly and indirectly test requirements specification \( R \)
3. List of requirements specifications that are directly tested by test case \( T \)
4. List of requirements specifications that are directly or indirectly tested by test case \( T \)
5. List of requirements specifications that are directly tested by at least one test cases
6. List of requirements specifications that are directly or indirectly tested by at least one test cases
7. List of requirements specifications that are not tested

Here we say that a test case \( t \) directly tests a requirements specification \( r \) if \( t \) and \( r \) have similarity degree equal to or greater than the threshold \( X \). Also we say that a test case \( t \) indirectly tests a requirements specification \( r_1 \) if \( r_1 \) and another requirements specification \( r_2 \) have the similarity degree equal to or greater than the threshold \( Y \) and \( r_2 \) is directly tested by at least \( \alpha \) test cases including \( t \).

In the following of the paper, we set \( X = 0.5 \), \( Y = 0.7 \), and \( \alpha = 2 \) and hard-code these values in queries for presentation simplicity.

**4.2. Queries for retrieval of coverage information**

For each of the above items, we created a Cypher query and an SQL query to extract it. Due to the limit of space, we only present the queries for the first and sixth items. Note that the $s and %s in the queries serve as formal parameters and are replaced with actual arguments at runtime.

Query 1, the query for the first item, receives the name of a requirements specification as an actual argument and
returns a list of test cases that directly test that requirements specification. When the query is executed with \( r1 \) being the actual argument for the running example shown in Section 3.1, the query should return two test cases, \( t1 \) and \( t7 \).

Below we show two queries written in Cypher and SQL. From the description of the query in Cypher, it is seen that the Cypher query very succinctly describes the pattern that matches relationships the target test cases must possess. On the other hand, the SQL implementation of Query 1 is slightly more complicated because it requires joining the node tables and the edge tables in order to obtain the similarity between test cases and the requirements specification.

### Cypher query 1

1. MATCH (r:Requirement)-[s:Similarity]-(t:Testcase)
2. WHERE r.name = $s AND s.value >= 0.5
3. RETURN t.name AS test_name
4. ORDER BY test_name

### SQL query 1

1. SELECT test.name AS test_name
2. FROM edge_req_to_test edge
3. JOIN node_req req ON edge.from_id = req.id
4. JOIN node_test test ON edge.to_id = test.id
5. WHERE req.name = %s AND edge.similarity >= 0.5
6. GROUP BY req.id
7. HAVING count(*) >= 1
8. UNION
9. SELECT req.name AS req_name FROM (SELECT id1, id2 FROM (SELECT edge.from_id AS id1, edge.to_id AS id2 FROM edge_req_to_req edge WHERE edge.similarity >= 0.7 UNION SELECT edge.to_id AS id1, edge.from_id AS id2 FROM edge_req_to_req edge WHERE edge.similarity >= 0.7)) AS rr JOIN edge_req_to_test rt ON rr.id2 = rt.from_id JOIN node_test test ON rt.to_id = test.id WHERE rt.similarity >= 0.5 GROUP BY rr.id1, rr.id2 HAVING count(*) >= 2)
10. UNION
11. SELECT req.name AS req_name FROM (SELECT edge_req_to_test edge FROM edge_req_to_test WHERE edge.similarity >= 0.5 GROUP BY edge.to_id ORDER BY edge.to_id)
12. ORDER BY test_name;

Query 6 for the sixth item obtains a list of all requirements specifications that are tested directly or indirectly by at least one test case. For the running example, the query yields a set of five requirements specifications: \( r0 \), \( r1 \), \( r2 \), \( r4 \), and \( r5 \). In the Cypher implementation of this query, even more complex pattern matching than Query 1 is described in an intuitive way. On the other hand, the SQL query is long and difficult to understand, because many JOIN operation are performed and similar descriptions need to be repeated for matching requirements specifications.

### Cypher query 6

1. CALL {
2. MATCH (r:Requirement)-[s:Similarity]-(t:Testcase) WHERE s.value >= 0.5
3. RETURN r.name AS req_name
4. UNION
5. MATCH (r1:Requirement)-[s1:Similarity]-(t:Testcase) WHERE s1.value >= 0.5
6. WITH r1, count(t) AS count WHERE count >= 2
7. MATCH (r1:Requirement)-[s2:Similarity]-(r2:Requirement) WHERE s2.value >= 0.7
8. RETURN r2.name AS req_name
9. }
10. RETURN req_name
11. ORDER BY req_name

### SQL query 6

1. (SELECT req.name AS req_name
2. FROM edge_req_to_test edge
3. JOIN node_req req ON edge.from_id = req.id
4. JOIN node_test test ON edge.to_id = test.id
5. WHERE edge.similarity >= 0.5
6. GROUP BY req.id
7. HAVING count(*) >= 1
8. UNION
9. SELECT req.name AS req_name FROM (SELECT id1, id2 FROM (SELECT edge.from_id AS id1, edge.to_id AS id2 FROM edge_req_to_req edge WHERE edge.similarity >= 0.7 UNION SELECT edge.to_id AS id1, edge.from_id AS id2 FROM edge_req_to_req edge WHERE edge.similarity >= 0.7)) AS rr JOIN edge_req_to_test rt ON rr.id2 = rt.from_id JOIN node_test test ON rt.to_id = test.id WHERE rt.similarity >= 0.5 GROUP BY rr.id1, rr.id2 HAVING count(*) >= 2)
10. UNION
11. SELECT req.name AS req_name FROM (SELECT edge_req_to_test edge FROM edge_req_to_test WHERE edge.similarity >= 0.5 GROUP BY edge.to_id ORDER BY edge.to_id)
12. ORDER BY req_name;

Table 2 compares the Cypher and SQL queries in terms of character count. The queries in SQL are approximately 1.5 to 1.9 times longer than those in Cypher.

### 5. Case study

In this section, we describe the results of executing the queries shown in the previous section. For the experiment, we obtained traceability data by analyzing artifacts documented in a real-world project. The dataset is in a text format and contains 3,855 test cases and 260 requirements specifications.

The experiment was conducted on a Windows 10 PC
Table 2: Number of characters in each query

<table>
<thead>
<tr>
<th>Query type</th>
<th>Cypher</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>116</td>
<td>179</td>
</tr>
<tr>
<td>Query 2</td>
<td>415</td>
<td>724</td>
</tr>
<tr>
<td>Query 3</td>
<td>114</td>
<td>177</td>
</tr>
<tr>
<td>Query 4</td>
<td>413</td>
<td>795</td>
</tr>
<tr>
<td>Query 5</td>
<td>110</td>
<td>192</td>
</tr>
<tr>
<td>Query 6</td>
<td>325</td>
<td>599</td>
</tr>
<tr>
<td>Query 7</td>
<td>514</td>
<td>639</td>
</tr>
</tbody>
</table>

with an AMD Ryzen 5 3600 CPU and 16GB of memory. We used Neo4j graph database management system and PostgreSQL relational database management system.

To load the traceability data into these databases, we implemented Python scripts which parse given data and call database APIs to update the databases accordingly. This process required about 47 minutes for Neo4j and about 4 minutes for PostgreSQL.

Using the databases loaded with the traceability data, we measured the processing time of the queries. Table 3 shows the processing time for each query. Since Queries 1 to 4 take a test case or a requirements specification as input, we measured the processing times for all test cases or requirements specifications and averaged them.

For Neo4j, the longest execution time was observed when Query 7 was executed. This query obtains the list of specifications that are not tested at all. Considering the fact that the query has to exhaustively check all requirements specifications, the processing time, which was about 13 seconds, is sufficiently permissible. In addition, Queries 1 to 4, which are queries concerning a specific requirements specification or test case, were all executed in less than 0.6 seconds. For PostgreSQL, on the other hand, the longest execution time was observed when Query 4 was executed; but the time was only approximately 0.1 seconds. Postgresql exhibited shorter processing time than Neo4j for all cases.

Table 3: Execution time (seconds)

<table>
<thead>
<tr>
<th>Query type</th>
<th>Neo4j</th>
<th>PostgreSQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query 1</td>
<td>0.004</td>
<td>0.056</td>
</tr>
<tr>
<td>Query 2</td>
<td>0.054</td>
<td>0.075</td>
</tr>
<tr>
<td>Query 3</td>
<td>0.002</td>
<td>0.054</td>
</tr>
<tr>
<td>Query 4</td>
<td>0.047</td>
<td>0.125</td>
</tr>
<tr>
<td>Query 5</td>
<td>0.359</td>
<td>0.048</td>
</tr>
<tr>
<td>Query 6</td>
<td>5.086</td>
<td>0.098</td>
</tr>
<tr>
<td>Query 7</td>
<td>13.440</td>
<td>0.098</td>
</tr>
</tbody>
</table>

6. Related work

Studies that consider the applicability of graph queries in the area of software engineering include [4, 5, 6]. In reference [4], four query languages including SQL and Cypher were compared for test case traceability queries. The results show that Cypher is superior in terms of expressiveness and understandability. Although their work and ours both concern the applicability of graph queries, the sorts of data and the purposes of using the queries are significantly different. For example, [4] considers traversal of traceability links whereas ours concerns the coverage of requirements specifications by test cases. Reference [5] uses a network to represent the traceability links between requirements, code, and test cases and compares the conciseness of the SQL and Cypher representations of two types of simple queries. Their work also considers traversal of traceability links; they did not deal with the kind of coverage analysis we did. Reference [6] analyzed the performance of query processing for large-scale software artifact data: it is shown that querying a relational database running on Spark, which is a cluster computing framework, with SQL is more efficient than using Neo4j and Cypher. On the other hand, in this paper, we showed that even querying Neo4j running on a single computer exhibited sufficiently practical processing time for a real-world data set. The result does not conflict with ours, where PostgreSQL exhibited better performance than Neo4j on a single computer; but our results also show that graph queries can be executed sufficiently fast for a real-world data set.

In [3, 7] we developed an approach to automatically find traceability links between test cases and requirements specifications. This approach first estimates the similarity degree between two artifacts using natural language processing techniques and then infers the existence of traceability links using the estimates. The dataset used in the case study of this paper was obtained using the first step of this approach. The problem of measuring similarity or relevance between software artifacts have also been studied elsewhere, especially in the context of automatic construction of traceability links between artifacts. Examples of this line of studies include, for example, [8, 9, 2, 10, 11].

7. Threats to Validity

A major internal threat to validity stems from the expressiveness of Cypher and SQL. In general queries in these languages have different representations: thus the queries we presented here might have more intuitive or concise alternatives. Another threat lies in how to compare the conciseness of the queries. In this paper we measured the conciseness in terms of character count; but another measure, for example, the number of tokens, might be more appropriate.
An external threat of validity concerns the representativeness of the data we used in the case study. Although the data was obtained from one of the largest projects lead by our industrial partner, there should be projects that need to manage data of larger size. Other characteristics of data, especially, the distribution of similarities can also vary from projects to projects. In view of these, a care should be taken when generalizing the findings about query processing performance.

8. Conclusion

In this paper we discussed the applicability of graph databases and graph queries to coverage analysis between test cases and requirements specifications. We showed that traceability data can be represented as a graph database and that coverage information can be easily retrieved from the database with graph queries. We also demonstrated that the graph queries, which are written in the Cypher query language, are often more concise than those in SQL, while the processing performance is comparable between the graph database and the SQL database.

Future research include many possible directions. The case study of this paper concerned a single product. Data sets from other systems, particularly larger ones, should be considered in future. Extending the list of queries for coverage analysis also deserves further research. To this end we plan to interview developers from industry to find out other queries that are useful in practice.

Acknowledgment This work was supported by JSPS KAKENHI Grant Number JP20K11747.

References


Using the Normalized Levenshtein Distance to Analyze Relationship between Faults and Local Variables with Confusing Names: A further Investigation

Carmine Gravino* Alessandra Orsi* Michele Risi*

* Department of Computer Science
University of Salerno, Italy
{gravino, mrisi}@unisa.it

Abstract

This paper exploits further uses of NLD (Normalized Levenshtein Distance), proposed in a recent study, to quantify the level of confusion of variables with the aim of verifying if they can provide indications about the presence of faults. We provide further evidence that fault prediction models based on the considered NLD measures can provide accurate estimations.

1 Introduction

This paper presents a further investigation about the use of the Normalized Levenshtein Distance (NLD) proposed by Tashima et al [1]. NLD allows to quantify the string similarity between local variables by exploiting the Levenshtein distance. In particular, it determines the minimum number of modifications in the characters to change one string to another string. NLD allows to quantify the confusion of local variables [1] and verify if the presence of not easily distinguishable variables in a method can suggest that the method is fault prone. In addition, we propose further uses of NLD: NLD$_1$, which represents the sum of all the NLD values calculated for each pair of variables declared in a method; NLD$_2$, which represents the average of all the NLD values for each pair of local variables; and NLD$_3$, which is defined as the sum of all the NLD values for each pair of local variables multiplied by the number of LOCs of the method.

To assess all the considered confusing measures, we have performed an empirical study by considering the same software systems employed in [1]. The results confirm and extend the ones of previous study about the relationships between the presence of faults and local variables with confusing names and how fault prediction models (built using the Random Forest) based on the considered distances can provide accurate estimations.

Organization of the paper: In Section 2 we summarize related work and recall the definition of NLD [1]. The design and the results of the performed empirical study are presented in Section 3 and Section 4, respectively. Conclusion concludes the paper.

2 Background

2.1 Related work

Software fault prediction has been widely investigated, aiming at identifying source code information that can help to accurately predict the presence of faults (e.g., [2] [3]). A few papers have investigated how the choices of developers when naming local variables can impact software quality. We can start mentioning the indications provided by Kernighan and Pike [4], who state that local variables have restricted role being used in a limited scope, thus it is unnecessary to use long and descriptive names for these identifiers. Some years later, Lawrie et al. [5] have performed a quantitative survey to investigate the impact of the variable composition on software comprehension. In their analysis they take into account three types of identifiers: a fully-word, abbreviated word, and a single character. The analysis of results reveals that understandability of identifiers decreases from full-words to single-character words. However, no significant difference can be highlighted between the use of full-words and abbreviated identifiers in terms of source code comprehension. More recently, a large-scale experiment performed by Scanniello et al. [6] has achieved similar results. They conducted a qualitative study to understand how identifier names either abbreviated or full-word the values, impact on fault fixing. Furthermore, it seems that even if variables with long names can help to better understand their use, the overall source code readability can be reduced [7]. Another study by Aman et al. [8] has also revealed that long local variable names are change-prone.

Binkley et al. [9] performed a study to compare the impact on the program comprehension when programmers use different naming styles. They compared the use of the camel case against the use of the snake case. The results show that the camel case improves the source code comprehension for developers at the beginning of their career while there is no significant difference for expert developers.

Regarding the relationship between the naming style and the presence of faults, the findings of a study performed by Kawamoto and Mizuno [10] reveal that the source code results to be fault-prone when classes contain long identifier names. With the aim of showing good practices when naming the identifiers, Butler et al. [11] have defined and as-
sessed 12 naming rules. The results of the performed analysis show that the use of identifiers not following the proposed rules increases the presence of faults.

Differently from the above mentioned contributions, Tashima et al. [1] have recently focused their attention on pairs of local variables with similar and confusing names. The aim of their investigation is to verify the relationships between the presence of such confusing variables and the fault-proneness at method level.

2.2 Normalized Levenshtein Distance

The motivation of Tashima et al. [1] is that the presence of local variables with high similarity names implies the possibility to confuse their use in the source code. To this aim, the Levenshtein distance is used to evaluate how much two names are confusing. The Levenshtein string edit Distance (LD) algorithm is one of the most important models for string matching [12]. This edit distance is defined as the minimum number of insert, delete, and replace operations required to transform a source string $x$ into a target string $y$. The approach assumes that insert and delete operations have cost 1, while the replacement has cost 2 (it is equivalent to a sequence of delete and insert operations). However, to evaluate more precisely the degree of confusion between two strings (local variables’ names) $s_1$ and $s_2$, Tashima et al. propose a normalized LD, which is computed by dividing the distance by a factor that depends on the length of the considered local variables:

$$\text{NLD}(s_1, s_2) = \frac{\text{LD}(s_1, s_2)}{\max\{\lambda(s_1), \lambda(s_2)\}}$$

where $\text{LD}(s_1, s_2)$ is the Levenshtein distance between the two strings $s_1$ and $s_2$, and the function $\lambda$ computes the length of the corresponding string. In particular, Tashima et al. adopted the following definition:

$$\text{NLD}_T(s_1, s_2) = \min_{\forall s_1, s_2 \in M, s_1 \neq s_2} (\text{NLD}(s_1, s_2))$$

3 Study design

We have formulated the following research question:

**RQ** Can information on variables with confusing names help to predict the presence of faults?

To answer RQ we have built different prediction models based on the considered distance measures and assessed their accuracy in prediction. We also decided to build a prediction model exploiting the Line of Code (LOC) metric as independent variable, to verify whether the predictions achieved with NLD based measures are better than those obtained using only LOC.

3.1 Exploited NLD based measures

We considered further uses of NLD proposed in [1], starting from two considerations: Why lower confusion values are excluded? Can the use of other software size measures (like the size of the module being analyzed) improve the effectiveness of NLD? To this aim, we consider three further NLD based measures:

- NLD$_1$: a “cumulative” measure computed by simply adding all the values of NLD for each pair of variables (i.e., we performed all the possible permutations of the identifiers defined in a method):

$$\text{NLD}_1 = \sum_{s_1, s_2 \in M, s_1 \neq s_2} \text{NLD}(s_1, s_2)$$

where $s_1$ and $s_2$ can be all the local variables defined in a method $M$ of the analyzed software class.

- NLD$_2$: it is based on NLD$_1$. In particular, the cumulative value of all the obtained distances is normalized by a factor depending on the number of all the local variables in the method $M$: $\text{NLD}_1/n$, where $n$ is the overall number of local variables defined in $M$.

- NLD$_3$: it is obtained by multiplying the NLD$_2$ value by the number of LOCs present in the method under consideration (i.e., $\text{LOC}_M$): $\text{NLD}_2 \times \text{LOC}_M$

3.2 Datasets

We considered the same five open source projects employed by Tashima et al. [1] for different reasons. We were interested in analyzing software implemented in Java and managed with Git in order to identify useful information such as the presence of faults. And more important, we selected the same software since our aim was to further assess the accuracy of NLD based measures (including NLD$_T$) given that NLD$_T$ provided good results on these software as reported in the original work of Tashima et al. [1]. In particular, the systems are: Apache Tomcat v. 9.0.12, Birt v. 4.8.0, Eclipse JDT User Interface v. 4.10.0, Eclipse Platform User Interface v. 4.10.0, Eclipse SWT v. 4.9.

In order to conduct the study, it was necessary to collect data from different sources. In particular, the collection of information to calculate the confusing measures, i.e., NLD$_T$, NLD$_1$, NLD$_2$, and NLD$_3$, was computed by analyzing the local variables of the methods of the source code of the considered projects. To this aim, we exploited a parser written in Java that makes use of the Eclipse JDT core library to extract information on methods and their local variables. We computed the values for NLD for each pair of local variables and then the values of each NLD$_i$ ($i \in \{T, 1, 2, 3\}$) as described above. Then, we add data about the presence of faults for each method by exploiting information from Promise repository [13], also used by
Tashima et al. but for different versions of the software projects, and by manually analyzing information provided in Git. This was the only strategy to adopt since the versions of the five projects we considered are among the most recent and are not the same as in the previous study by Tashima et al. [1]. Thus, the fault recovery was made by making an intersection at method and class levels between the datasets used by Tashima et al., containing also the faults, and those used in our study. Whenever there was a correspondence of modules between the old and the new versions of a given project, the fault related to that module in our dataset for the project was added. In particular, in our analysis NLD$_T$, NLD$_1$, NLD$_2$, NLD$_3$ have been used as independent variable while Fault (the presence of fault, i.e., 1, or not, i.e., 0) as dependent variable.

### 3.3 Prediction models and accuracy evaluation

To build our fault prediction models, we employed the Random Forest that is a popular method for various machine learning tasks. It exploits a classifier as specified above, however it constructs more classification trees instead of a single tree [14]. As for its implementation, we exploited the tool Weka that offers widely used estimation techniques [15]. In particular, for our analysis we used the Classify module by selecting: i) all the parameters necessary for the construction of the model, ii) the Random Forest algorithm, iii) the independent variable (i.e., a measure of confusion), iv) the dependent variable (Fault) and the type of the validation method to be used.

To verify whether or not the obtained fault estimations are useful predictions of the actual faults we exploited 10-fold cross validation [16] with $k = 10$, which requires the splitting of the dataset in $k - 1$ training sets and 1 validation test for $k$ times. Each time the training set is employed to define the estimation models, with the selected estimation techniques, while the corresponding validation test is used to validate the predictions obtained with the built models.

To evaluate the accuracy of the fault predictions, we employed F-measure defined as the weighted harmonic mean of the Precision and Recall [17]. Since the fault estimations have been computed on a dependent variable representing two classes with a very different number of observations, the Matthews correlation coefficient (MCC) can be generally adopted to measure the quality of a binary classifier. MCC represents a correlation coefficient between the observed and predicted binary classifications [18]. The MCC measure ranges from $+1$ for a perfect classifier through 0 for a random classifier to $-1$ for a weak classifier.

### 3.4 Threats to Validity

Some threats could affect the validity of our analysis. We considered five software projects developed in Java, and so the number and type of software can introduce a bias with respect to external validity. Thus, further investigations with different type of software projects and a greater number of projects should be carried out. However, to mitigate this threat we considered software projects whose information are publicly available and employed in previous investigations. Regarding the collection of information we employed Eclipse JDT to analyze source files to calculate the confusing measures, by analyzing the local variables of the methods of the source code of the considered projects. Eclipse JDT is a widely used tool for accomplishing such kind of work. As for the collection of fault data, possible threat is related to the fact that the fault recovery was made by making an intersection at method and class level between the datasets of the Promise used by Tashima et al. [1], containing also the faults, and those used in our study.

Other threats can regard the data analysis performed. As for the technique applied to obtain the prediction model we exploited Random Forrest since it is widely used for classification problems similar to ours. Furthermore, it was also used in the original work by Tashima et al. [1]. As for the assessment of the achieved fault predictions, other measures could be used, such as accuracy, however F-measure and MCC are widely employed in studies similar to ours.

### 4 Results

First of all, for two of the confusing measures, i.e., NLD$_1$ and NLD$_3$, we have observed a specific relationship between the presence of faults and local variables with confusing names, namely, as the value of confusing measures increases (i.e., the distance between local variables increase and so they are less confusing) the value of the fault rate
increases as well. For the other two considered confusing measures we cannot provide a clear trend.

Table 1 reports the results in terms of Correctly classified instances (%), Incorrectly classified instances (%), F-measure, and MCC for each software systems and the built estimation models (i.e., based on NLD₁, NLD₂, NLD₃, NLD₄, and LOC) obtained by averaging the results of the 10-fold cross validation as designed in Section 3.3.

We can note that F-measure values range from 0.5 to 0.85. The greatest values were obtained with Apache Tomcat (which is smaller in size with respect to the others) while the worst values were obtained with Platform ui. A similar consideration can be provided for MCC values.

The NLD based measures used as independent variables in fault prediction models built with the Random Forest that allowed to obtain better predictions are NLD₁ and NLD₃. NLD₄ provided results similar to NLD₁ and in one case (i.e., Apache Tomcat) provided better predictions than the others. For the other 4 systems the measure that allowed to obtain better predictions is NLD₃. Let us remember that NLD₁ is the sum up NLDs of all the pairs, thus as a long method or a complicated method tends to have more variables its NLD₁ tends to be larger. NLD₃ is NLD₂ multiplied by LOC, so greater LOC greater NLD₃. So, both NLD₁ and NLD₃ are influenced by the source code size and one can image that the size measure has a large influence on the defect prediction performance. However, we can note from Table 1 that the good results in terms of NLD₃ for the systems JDT ui and Apache Tomcat are mainly due to the contribution of NLD₂, which is not related to size, or to the interaction between size and the confusion of variable (i.e., NLD₂). In two cases (i.e., Birt and Platform ui) LOC seems to have contribute more to the results achieved in terms of NLD₃. Moreover, it is important to note that NLD₁ allowed to obtain better results than LOC for all the software systems. In particular, for JDT ui, SWT, and Apache Tomcat the predictions achieved with NLD₁ and NLD₃ are particularly better than those obtained with LOC. Thus, we can positively answer RQ2 because information on variables with confusing names can help to predict the presence of faults. However, given that the our study is conducted on source code developed in open-source projects, our answer is cautious though. Indeed, observe that predictions are more accurate on some systems (i.e., Apache Tomcat) than on others independently from the confusing measures used.

5 Conclusions and Future Work

Our results and those of the original work by Tashima et al [1] can provide evidence of the usefulness of knowledge of variables with confusing names to improve the quality of the source code. In the future we intend to further investigate the relationship between faults and local variables with confusing names by considering different datasets and other combinations of NLD based and source code measure.

Acknowledgment

The authors would like to thank Hirohisa Aman [1] for the suggestions provided for the identification of the case studies used in the empirical analysis.

References


A Framework for Mutation Testing of Machine Learning Systems

Raju Singh
Department of Computer Science & Information Systems
BITS Pilani
Pilani, India
p20200106@pilani.bits-pilani.ac.in

Mukesh Kumar Rohil
Department of Computer Science & Information Systems
BITS Pilani
Pilani, India
rohil@pilani.bits-pilani.ac.in

Abstract—In this paper, we provide an insight journey of Testing of Machine Learning Systems (MLS), its evolution, current paradigm, and we propose a machine learning mutation testing framework with scope for future work. Machine Learning (ML) Models are being used even in critical applications such as Healthcare, Automobile, Air Traffic control, Share Trading, etc., and failure of an ML Model can lead to severe consequences in terms of loss of life or property. To remediate this, the ML community around the world, must build highly reliable test architectures for critical ML applications. At the very foundation layer, any test model must satisfy the core testing attributes such as test properties and its components. These attributes should come from the software engineering discipline but the same cannot be applied in as-is form to the ML testing and in this paper, we explain why it is challenging to use Software Engineering Principles as-is when testing any MLS.

Keywords—Machine Learning, Software Testing, Quality Attributes, Deep Learning, Model Mutation Testing.

I. INTRODUCTION

In the current context of software development and machine learning (ML), it is inevitable, not to come across an ML scenario in day to day life. It spans across business critical applications such as share trading, insurance and banking, medical applications such as drug manufacturing, identification of disease and medical imaging, and safety critical applications such as autonomous driving and robotics [1]-[3]. Software testing [4] of mathematical software [5] and Intelligent systems [6] has adopted some of the software testing methodologies. Applications of ML in several critical sectors make ML testing [7] a reliable way to ensure quality and minimize failure scenarios. An adoption of testing framework from traditional software testing with the addition of key ML quality attributes [7] makes more sense. Testing framework that covers performance (for critical real time systems), security (for business applications and health care applications) and safety (for system of systems) increases its trustworthiness.

To better understand the testing challenges for ML systems [8]-[10], we need to deep dive as how ML systems are different from traditional software system. Traditional software is more deterministic in nature, lacks dynamicity in terms of varied inputs. On the other hand, ML systems are dynamic, non-deterministic and expected to learn from data (labels), and predict the output accordingly.

For instance, a rover has to determine the path on a rocky terrain based on the imaging data that it gathers from the surrounding, the forest fire alert systems have to generate a prediction based on the environmental data such as air humidity, wind, temperature and climatic conditions. The model tends to evolve and learn from historical data.

Oracle Problem [11]: Machine learning models are difficult to test because they are designed to solve problem based on learning from past experience (label data, supervised learning), without past experience (unsupervised learning) or through re-enforcement. Attempts have been made to draw parallel between the ML testing approach with Software Testing. By understanding the process of software development, we should be able to break down the software stack into components (unifiable units), and build test cases around it. In other approaches, we have Test Driven Development (TDD) to setup in testing framework. This approach might not work well with ML models. It is because, machine learning models are mostly monolith, and components may not reflect the true nature of the ML model as a whole. Again, breaking ML model into unifiable components or developing with TDD is a cumbersome task.

In order to understand and design a test framework for ML system, we need to understand the behaviour [4] of the model, and how the model interacts with the surroundings. Studying behaviour of the ML model, gives limited insight into the model.

Further the paper is organized as: Section II summarises the background and related work, Section III summarises Machine Learning Testing Scenarios in terms of faults, failures and Oracle problem, Section IV explains machine learning testing by considering dynamicity of MLS, Section V describes proposed framework for machine learning system testing, Section VI describes prediction mutation testing to explore the possibility of testing machine learning models such as Deep Neural Network (DNN) models, Section VII summarizes model mutation testing, Section VIII presents the major challenges, Section IX lists the assumptions and hypothesis for set of transformation rules to yield mutated DNN models, Sections X brings out some approaches for mutation testing, and Section XI furnishes conclusion and scope for future work.

II. BACKGROUND AND RELATED WORK

Mutation testing in traditional software predates any similar testing framework in Machine Learning and DNNs specifically. Here, the mutation testing is a proven tool and has higher
accuracy. Mutant operators form a well-researched area that has its implementation in several high level programming languages for traditional software. With time and increasing complexity of traditional software, the mutation testing framework has been extended well. Use of mutation testing in machine learning, is being extensively researched and several researchers have established milestones for most of the known DNN models. One such approach is DeepMutation.

**Machine Learning**: Machine learning is a field of study that gives the ability to computers to learn without being explicitly programmed. A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E [12], [13].

Machine learning is a phased approach. The first phase is the learning phase. In this phase, data is gathered and bucketed as training and test data sets. Training data set is identified by attributes and label. The outcome of this phase is a model that is drawing the relationship between the attributes and the label. The subsequent phase deals with applying the model to different dataset (test data). There are several algorithms to accomplish this, such as classification algorithms, ranking algorithms, etc. There are several attempts and general-purpose availability of model based mutation testing, which depends upon comparing results from different test scenarios.

Terms used in machine learning domain:

**Dataset**: An ingredient for machine learning model, consists of sets of instances for building or evaluating the model. It is further categorized as:

**Training Data**: This data is obtained from the sources (sensors, data collection devices, etc. aggregated and cleaned up to exclude bias and noise) and is used for the purpose of training a machine learning model. This model is basically an organic algorithm which learns from the training data and performs a particular task.

**Validation data**: This data is from the training data, used to tune the hyper-parameters of learning algorithm.

**Test data**: This data is the part of training data, for which machine learning model has not been trained yet. Based on the performance of the ML model and its behaviour with the test data, we can attribute the machine learning model maturity.

Sub Definitions:

**Instance** is an information record about the object. **Feature** is a measurable property. Errors are also an important aspect of machine learning, and it is this property that the model behaviour depends on. **Test error** is mainly focused on deviation measure between the obtained value and the expected value.

Let us classify Machine Learning.

**Supervised Learning** [9]: The goal is to predict the value of an outcome measure based on the number of input measures. It is commonly referred to as regression [14] problem since its outcome measurement is quantitative.

**Unsupervised Learning** [13]: The goal is to describe the association and patterns in a set of input measures. We only observe the feature and have no measurement about the outcome.

**Reinforcement Learning** [16]: In this approach, agent (learning system) can observe an environment, selectively perform actions, and get rewards (or penalties). The key here is, it must learn by itself, and accordingly respond in actions for the good. This approach is referred to as policy, to get most reward over a period of time. In nutshell, a policy defines what actions the agent should take when subjected to a condition.

### III. Machine Learning Testing Scenarios

Fault and Failures [17], [18]: The following discussion involves classification of the faults and failure scenarios of the ML systems. Since most of the ML systems deal with uncertain components, faults and failures are possible in ML models. These can be handled by creating counter measures to prevent failure scenarios, however, we can have inevitable scenarios.

Definitions in the IEEE Standard Glossary (IEEE 1990) [19]:

**Fault**: An incorrect step, process, or data definition in a computer program.

**Failure**: The inability of a system or component to perform its required functions within specified performance requirements.

**Data Sensitive Fault**: A fault that causes a failure in response to some particular pattern of data.

**Program Sensitive Fault**: A fault that causes a failure when some particular sequence of program steps is executed.

The core of the testing system is to find the deviation of ML models from the expected outcome.

**Oracle**: Oracle tests are basically intended towards the Behaviour test. This is a challenging aspect as the behaviour of ML systems is unpredictable, and this unpredictability makes sense to build oracle tests. In MLS context, Metamorphic Oracles have gained ground as a feasible approach to infer oracle information from data. Metamorphic oracles insight metamorphic relations between input values, i.e. if a metamorphic relation exists between the inputs, the corresponding MLS outputs must satisfy a pre-existing relation (ideally, equality or equivalence relation). Input data and its dimension poses a greater instability towards the ML testing realm. So, it is vital to choose adequate test data in order to cover impactful dimensions.

### IV. Machine Learning Testing

Software testing techniques, such as unit, integration and system testing [20], can be used in ML testing domain. Additionally, in order to address the dynamicity of the ML systems, additional recommendation has been made for ML testing domain which includes input, model and integration testing.
Input Testing: These tests are concentrated on the input data which is used to train the ML model. The core reason of using input test is to minimise the risk of faults. It can be either offline testing or online testing. During the offline testing, it detects faults by alerting the bias in the training data. In online testing, where ML models are expected to predict for unlabelled data, this testing helps on input validation.

Model Testing: Model testing tests the function aspects of the system under test (SUT) (ML model in isolation, without taking any other component into account). It tries to find the faults in the model architecture, training process, etc. It uses accuracy (for classifier) or mean squared error (for regressions [14], [21]). It is sometimes considered as unit testing.

Integration Testing: Integration testing considers the integration aspect of ML models, hardware systems, software systems and their interactions.

System Testing: System testing is a holistic test to evaluate the system’s measures under a given requirement.

Black-box and White-box Testing: Black-box testing [22] screens the internal structure of the design, code and its implementation, of ML systems, without having access to the core while white-box testing is crucial as it knows the internal structure of the code, design, implementation and behaviour of the ML systems. This way it makes more sense to use white-box testing in ML model. In contemporary software, source code is the main source of faults or defects. Mutation testing injects modified program code to introduce defects or faults, and this enables the qualitative measurement of test data by detecting manual changes. With the knowledge on such mutation testing framework, we can suggest a Deep Learning (DL) based mutation testing framework with two stage process.

Source level mutation: DL systems depend on the training program and training data. Training process is defined as the articulation of training program on training data. The master source code of training program and master record of the training data is mutated over a period of time during the testing and the deviation of the Model is recorded. The new evolved model, result from the mutation exercise, is set to run through the training set in order to determine the quality of the test data. The mutation operator can be categorized as: data mutation operator and program mutation operator.

Data Mutation Operators: DL model depends heavily on training data. We know that DL model’s robustness depends on the underlying data quality. Error introduces at any stage of data collection, data aggregation and data cleaning, and skews the DL model as the data contains noise.

Program Mutation Operators: Training programs in DL systems are coded using high level languages, and use problem specific programming framework. Injection faults in the program would cause unexpected behaviour in the DL systems.

This requires us to carefully craft mutant operations to inject faults into the training program. The kind of fault we can think of now is like, addition and removal of layers from DL models, pass on skewed weights and activation function while training process.

Model Mutation Testing for DL Systems:

Most of the mutation testing frameworks which work efficiently in traditional software systems do not hold ground with the DL models. The problem is, most of the mutation testing from traditional system is written on the source code, or its low level representation such as byte-code. However, model mutation testing can be a better approach and we will show it.

In source level mutation testing, the algorithm injects modifications in the training data and training program, while in model level mutation testing, the algorithm updates the DL model obtained from the training program. As the expectation remains intact for both approaches, i.e. to evaluate effectiveness and weakness of the test data set, model level mutation testing leads the way forward by directly mutating the DL model.

ML System Attributes:

Security: ML systems are as vulnerable as any other software systems, along with few inherited vulnerability, given the model footprint. Security reciprocates to the robustness.

Efficiency: ML system efficiency reciprocates to accuracy of its prediction.

Fairness: ML systems suffer from statistical problems such as bias, deviations and skewness.

V. MACHINE LEARNING TESTING FRAMEWORK

Behaviour Framework: ML system might behave differently given similar data. The main challenge is to identify the extreme boundaries for a given input space. This is similar to boundary-value analysis.

Test Adequacy Criteria: Any test suite woven around an ML system, should satisfy the quality attributes. As the classical approaches (based on the source code control flow [23]) are not relevant to the ML systems, researchers are trying to find out new domain in order to satisfy the test adequacy.

Mutation: In contemporary software testing domain, mutation testing is gaining grounds. It has become an efficient tool to find the faults in the ML systems, by injecting mutants. DeepMutation fundamentally works at the model level, iterates through varying mutation within the boundary space.

DeepMutation: DNNs have gained ground in several critical applications such as healthcare, autonomous vehicle and robotics. Any DNN system can either be a Feedforward Neural Network (FNN) or a Recurrent Neural Network (RNN) system. An FNN system processes the input information at each layer and forwards it to the next layer. This process continues until the decision is reached. This way, the FNN model preserves the local properties of each layer. On the other hand, the RNN extends the Long-Short Term Memory (LSTM) or memory cells and partially propagates the information backward to secure temporal information of sequential inputs. This way, the decision at any stage not only depends on the given input, but also on the current state. This makes RNN reliable for handling
sequential data, for instance, Natural Language Processing (NLP). The spirit of mutation in DNN [24], [25] is similar to that of traditional software. The main idea behind DeepMutation is to introduce adequate number of mutants or operators. The mutants must satisfy the quality attributes for the testing framework, such as input (test) data analysis.

Traditional software is built upon decision logic. This logic is implemented in the form of program code, whereas the DL models and systems are guided by the underlying DNN structures and their weights.

The weight of DL system is generally obtained from executing training program on a given training data, and DNN structure is defined as the code of the training program. These are two potential reasons, a deviation in which cause behavioural issue in the DL systems. The mutation operator can be inflicted in either training data set of training program or both. Once the mutant operators are injected, training program is executed on training data to generate mutated DL models.

**DeepMutation Testing Framework:**

DNN uses high level languages such as Python and R, however DNN is represented as hierarchical data structure. We are going to shortly lay down on to discuss on the mutation testing framework for DL systems. The first step is to design source level mutation testing operators. These operators can modify the training data and training program. The basic idea behind this is to improve the data quality evaluation. The fault might be injected manually, or might naturally occur in the training data or in training program. This framework must address the mutated DL models efficiently and address issues such as computation resource requirements, security vulnerability issues. Given this, we must work backward to generate efficient mutant operators. Before we deliver further, we need to elaborate model-level testing.

**Model Level Testing:** A model is used to represent the desired behaviour of the system under test (SUT), or to represent the testing strategies and a test environment. A model representation of SUT is at abstraction or partial behaviour. We can derive only functional test cases from SUT. The idea here is to come up to the conclusion that how many model level mutation operators would result in the efficient generation of a set of mutations without inducing model level problem.

VI. PREDICTION MUTATION TESTING

Prediction Mutation Testing topic tends to attract two sorts of discussion – Mutation Testing approach in Machine learning, and – Machine learning approach in mutation testing. We will talk a little about the latter part, and then resume the discussion on the primary topic which is related to exploring the possibility of testing machine learning models such as DNN using mutation testing, its framework, challenges, pros and cons, future works, etc.

When we discuss the approach of machine learning and its impact on mutation testing, we consider that here mutation testing can be again applied to a software system or machine learning system. However, applying mutation testing in software system is inherently different from applying mutation testing in machine learning, and it is because of the behavioral changes that software systems and machine learning systems exhibit when subjected to mutation testing.

**Tradeoff - Efficiency versus Effectiveness:**

Any prediction model inherently suffers from accuracy problem. However, several experimental approaches in predication mutation testing domain have shown positive signs.
as it improves efficiency and accuracy of mutation testing. This is a clear indication of how prediction mutation testing stands out of traditional mutation testing. It will be worthwhile to look into the class probability distribution provided by the classifier, with which developer may choose the mutant with proper probability distribution in order to get better prediction result. It is a considerable improvement over traditional mutation testing, as it is light weight, in-expensive comparatively, with relatively high accuracy. In this article, it is assumed that mutation testing refers to prediction mutation testing.

VII. MODEL MUTATION TESTING

Models are common in software testing. It is used to select test suites. Applications of mutation testing at a model level can contribute to reliable and early assessment of the quality of the test suite. This can also help in defining a test suite which has high fault detection rates. One of the issues which we observed while using mutation testing [18] at the early development stage, is related to its reliability and quantifying it.

VIII. CHALLENGES

Contemporary coding for functional requirements is different from programming a DL model. The basic difference is, in contemporary programming, we break down the monolith requirements into small chunks of programmable units. Each unit is programmed separately and satisfies the software quality attributes such as correctness, fairness, security, etc. and then the units can be combined together with other modules to form the holistic program. In this approach, each unit or module has its own logic, an aggregation of which comply with the integrity attributes of the whole program.

While in DL systems, which are fundamentally data driven models, the logic that we can drive at the highest may be of abstraction. It might not be same when we try to modularize it. It is so because the logic is guided by the weight and activation functions. Moreover, DL systems are behaviour-driven systems which are built by executing training program on training data. Here, underlying logic is guided by the training data and not the requirement (as in traditional software).

IX. HYPOTHESIS

Let us make some assumption about the samples, adversarial samples and normal samples. In testing, adversarial samples are those samples which are vulnerable to any changes and show far more deviation in behaviour with respect to usual samples. Consider a scenario where the original DNN models have undergone a set of transformation rules to yield mutated DNN models. These mutated DNNs usually tend to label an adversarial data with a different label (label generated by original mutated DNN). We would assume this state, and try to measure the crucial factors such as model uncertainty estimate, density estimate, model sensitivity to the input changes.

Even before we can create a procedure for our hypothesis, we need an efficient way to generate the mutants. The fundamental approach is to generate or seed several program level mutations (mutants). This would require program under consideration to go through set of mutation operators by applying set of transformation rules. To the core of which lies, the process to define mutant operators. As it is known that traditional software systems are logic oriented, structured, while DNN models are behaviour and model oriented, therefore mutation operators’ application to the former scenario (Traditional Software Systems) does not work for the latter (DNN Systems). There are quite a few techniques which work independently and using mutation testing in order to establish the testing framework.

X. APPROACH

Initialization

One of the initial approaches would be a way to build foundation steps to measure Label Change Rate (LCR) for the adversarial samples and normal samples. This is measurable when we inject these samples into a set of already mutated DNN models.

<table>
<thead>
<tr>
<th>Mutation Operator</th>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian Fuzzing</td>
<td>Weight</td>
<td>Fuzzy weight by Gaussian Distribution</td>
</tr>
<tr>
<td>Weight Shuffling</td>
<td>Neuron</td>
<td>Shuffle selected weights</td>
</tr>
<tr>
<td>Neuron Switch</td>
<td>Neuron</td>
<td>Switch two neurons within a layer</td>
</tr>
<tr>
<td>Neuron Activation</td>
<td>Neuron</td>
<td>Change the activation status of a neuron</td>
</tr>
</tbody>
</table>

Let $x$: input sample (adversarial sample or normal sample). Let $f$: DNN model (post mutation operators are applied).

Now, we go through the model mutation operator as provided in the Table 1 (sequence wise) and select the mutation models. Quite a few times, the output mutated model is of moderate to low quality (assuming high precision and confidence as measure of high quality mutated models). This means that the accuracy and effectiveness on the training data work well, however on the test data, it significantly deprecates. We let go or ignore these low quality mutated models. Only mutated models with high accuracy are considered. We can adopt the scale based on our experience and historical data obtained from mutated models. Ideally, any model with more that 90% accuracy of the original model is part of the set. This is to make sure that we meet the decision boundary [25] conditions and they are not impacted much. Upon segregating the mutated models, we further obtain a label of the input sample on each mutated model.

Building a Model

In this stage, we follow the hypothesis to create a model. This model validates (on certain criteria) the observation. If we recount, earlier we mentioned that adversarial samples are generated in such a way that it tends to minimize the mutated behaviour on normal samples, while, it is being able to jump the decision boundary [25]. There are different ways of mutation to achieve this behaviour. As per the hypothesis, the effective adversarial samples are closer to the decision boundary. This minimizes the restricted modification in the model. With this,
adversarial samples would be considered as a case of crossing the decision boundary, unlike randomly selected mutated model. This implies, if we inject mutated adversarial sample into the mutated model, the outcome of the label tends to change it from its original label.

**Algorithm Design**

Experiments and test results show that LCR can be a distinguisher between adversarial samples and normal samples. We can discuss on the algorithm which can be designed to detect samples at runtime based on LCR measures of the provided samples. This algorithm would delete the LCR, and would keep on generating more effective and accurate mutated models. For this to happen, we must define a stopping condition on the mutation model generation algorithm that could be satisfied. Prediction algorithm can help us get a set of mutated models with higher accuracy beforehand.

**XI. CONCLUSION AND FUTURE WORK**

In this work, we proposed the machine learning mutation testing framework, its usefulness and approach to detect adversarial samples for DNN at runtime. We laid down the details of source level mutation techniques on datasets (training and test) and training (or test) programs. This required us to further the details of the process and techniques involved in designing source level mutation operators, and feed faults into the DNN models during their development and testing process. This accompanied the details of model level mutation technique. Model level mutation technique differs from the source level mutation technique in the approach that it adopts to inject the faults. Model level mutation technique directly feeds the faults into the DNN system. It is also noteworthy how to measure the quality of these mutation models.

We also briefly touched upon how to predict the mutant operators even before we can analyse the same by executing. This is primarily done as mutants’ generation is a computationally expensive approach. In the end, we proposed a hypothesis and an approach to build the problem set, analyse it and proceed under certain assumption to mitigate the same.

**REFERENCES**


A Case Study of Testing an Image Recognition Application

Chuanqi Tao①②, Dongyu Cao①, Hongjing Guo①, Jerry Gao①

① College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
② Ministry Key Laboratory for Safety-Critical Software Development and Verification, Nanjing University of Aeronautics and Astronautics, Nanjing, China
③ Department of Computer Engineering, San Jose State University, USA

Correspondence to: taochuanqi@nuaa.edu.cn

Abstract—High-quality Artificial intelligence (AI) software in different domains, like image recognition, has been widely emerged in our lives. They are built on machine learning models to implement intelligent features. However, the current research on image recognition software rarely discusses test questions, clear quality requirements, and verification methods. This paper presents a case study of a realistic image recognition application called Calorie Mama using manual and automation testing with a 3D decision table. The study results indicate the proposed method is feasible and effective in quality evaluation.

Keywords—image recognition; testing AI software; AI software quality validation

I. INTRODUCTION

With the rapid development of big data analysis and artificial intelligence technology, AI software and applications have been widely accepted in our daily life. At present, AI software and applications are based on the most advanced machine learning models, and various artificial intelligence features are realized through large-scale data training.

The most important implementation of Artificial Intelligence is the imitation of human interactions—vision. Nowadays, there is an abundance of digital images captured by high-quality equipment. Most images are captured with phones. Artificial Intelligence is often used to process these images to extract knowledge, categorization, and labeling along with other advantages. Typical applications of image recognition include object recognition, face recognition, text parsing.

Detecting bugs and errors in software can be very costly. Sometimes bugs can be even deadly if it is a real-time application of software, such as some software that is used to help with surgeries in the hospital. Therefore, testing the software is very important to verify that the product meets requirements and specifications. Software testing ensures the correctness, integrity, and high quality of the software by checking errors or bugs and fixing them in the initial design.

This paper focused on testing an image recognition application called Calorie Mama utilizing both manual testing and automation testing. Calorie Mama is a smartphone app that runs on Android and IOS devices. It uses deep learning to recognize food from food images and track nutrition based on the food in the image. It calculates the calorie based on that. We evaluated the performance, correctness, and quality of the app using both manual testing and automation testing.

This paper is written to provide our perspective views on image recognition software testing and quality evaluation. The paper is organized as follows. Section 2 discusses the review of AI software testing and image recognition. The third part shows a case study of testing Calorie Mama APP using manual testing and automation testing. Section 5 gives the conclusion finally.

II. RELATED WORK

Traditional software is implemented by developers with carefully designed specifications and programming logic. It is tested with test cases that are designed based on specific coverage criteria. However, the current practice of testing AI applications lags far behind the maturity of testing traditional software applications [1].

More and more work focused on testing AI-based software. Gao et al. [2] explained the various testing methods of AI software testing Various functional and non-functional quality parameters such as correctness, reliability, and scalability are discussed to better understand the concepts. Besides, the authors discussed the issues and challenges of AI testing. The different models of the AI system were discussed in [3]. The authors discussed building testable AI systems, limiting the AI system to propositional logic, and intervening variables in reducing testing. King et al. [4] discussed issues and challenges in software testing. They thought non-determinism is a huge issue. The same input to the system can produce different outputs. Testing has fuzzy oracles that determining the correctness can be a challenging task. Ramanathan et al. [5] used symbolic decision procedures coupled with statistical hypothesis testing to validate machine learning algorithms for studying the correctness of intelligent systems. They also used algorithms to analyze the robustness of a human detection algorithm built using the OpenCV open-source computer vision library.

In the field of image recognition, most of the researchers focus on recognition algorithms. Girshick et al. [6] proposed the R-CNN algorithm, which added selective search operations to the CNN network to identify candidate regions. He K et al. [7] proposed the SPP-Net algorithm, which reduced the process of image normalization and solved the problem of image information loss and storage. Girshick [8] proposed the Fast R-CNN algorithm, which refers to the Region of Interest and the multi-task loss function method, and replaces SVM classification and linear regression with Softmax and SmoothLoss to realize the unification of classification and regression and reduce the disk space.

However, the evaluation of the image recognition system is relatively less but important. In [9], the implementation of Yolo-v2 image recognition and other test benches for a deep learning accelerator was presented. Tao et al. [10] performed a case study...
on a realistic facial age recognition provided by Alibaba Company using metamorphic testing.

III. A CASE STUDY

A. Test Experiment

This paper took the test Calorie Mama APP as an example, using manual testing and automated testing respectively. The test data is a mix of various sources: images from Google, images clicked in real-time using a smartphone camera. The experiments were performed with a high-resolution and high-quality camera.

1) Manual Test

In this approach of manual testing, we selected conventional decision tables to test. A decision table is a table with various conditions and their corresponding actions. It is divided into four parts, condition stub, action stub, condition entry, and action entry.

1. Detection of non-food items: To test Calorie Mama, different non-food items are input into the application. The pictures of the non-food items were analyzed by the application and the results were shown on the user interface. A summary of the detection of the non-food items can be seen in the following decision table. The condition stub is designed as two conditions, including the state of the Internet and access to the Camera, which is essential for the image recognition software.

As we can see, the application detected artificial pumpkin and artificial cake as food items. In contrast, it could not correctly identify the butter block. As a result, it failed in some of the cases. Besides, when not turning on WIFI or Cellular, and not allowing access to the Camera, image recognition will not work.

<table>
<thead>
<tr>
<th>Food item</th>
<th>Wooden background</th>
<th>Egg in a bowl</th>
<th>Egg on a plate</th>
<th>Egg in the glass</th>
<th>Egg in a tray</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn on WIFI or Cellular</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Allow access to Camera</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

2) Detection of food items: We divided the generic term of food items into four categories which are Indian cuisine, raw fruits and vegetables, variety of apples and eggs, and food items in different backgrounds. Take food items in different backgrounds as an example, the background of food is a very important aspect and we decided to test the application with images of food items with different backgrounds.

As seen in table 2, the Calorie Mama application was able to correctly recognize the food items when given inputs with red, blue, and wooden backgrounds. However, the application detected wrong when the egg is in a tray.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
<th>R8</th>
<th>R9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn on WIFI or Cellular</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Allow access to Camera</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

After conducting the manual testing, we experienced its various drawbacks, and it is time-consuming. Also, load testing and performance testing are not possible under manual testing. Besides, regression test cases are very costly. Due to these drawbacks, we decided to shift to automation testing.

2) Data Modeling

The three-dimensional (3D) classification decision table is influenced by the concept of conventional decision tables to conduct classification-based test requirement analysis and modeling for any given mobile apps powered with AI functions using a 3D tabular view. The major testing focus for a 3D classification table is the mappings among classified disjoint context conditions, classified input selections, and classified AI function outputs. These mappings are known as image recognition function classification rules. Each of them represents the conjunction among three different views. Test case design and generation based on a 3D classification decision table must cover these image recognition classification rules. Adequate image recognition function testing coverage could be assessed. Next, we introduce the construction of each one-dimensional model in the 3D decision table.

1) Input Modeling

The input classification refers to the parameters and their values that represent the different test case scenarios. Each parameter has multiple possible values which when combined with context values gives us the final set of test cases. The following figure shows Calorie Mama's input classification tree, which contains information about the type of food being clicked, such as what the food is, and the physical appearance of the food, such as quality, size, shape, consistency, etc.

<table>
<thead>
<tr>
<th>Test Conditions</th>
<th>R1</th>
<th>R2</th>
<th>R3</th>
<th>R4</th>
<th>R5</th>
<th>R6</th>
<th>R7</th>
<th>R8</th>
<th>R9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Turn on WIFI or Cellular</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>Allow access to Camera</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Food item/Egg</th>
<th>Blue Background</th>
<th>Red Background</th>
<th>Wooden Background</th>
<th>Egg in a bowl</th>
<th>Egg on a plate</th>
<th>Egg in the glass</th>
<th>Egg in a tray</th>
</tr>
</thead>
<tbody>
<tr>
<td>right choices</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>wrong choices</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
2) **Context Modeling**

The context classification tree contains information about the image context. It is basic information about the image itself and not specifically about the item in the image. For example, the context classification tree contains information like if the image is blurry or not well illuminated, what is the angle of the camera while clicking the image, if the image is rotated or so, etc. The following figure shows Calorie Mama's context classification tree.

![Context Classification Tree](image)

Figure 2. Context Classification Tree

3) **Output Classifications**

The output classification tree contains information about the output. Various parameters regarding the output obtained from the application will be considered. This can be modified based on the requirements and results expected from the application. The following figure shows Calorie Mama's output classification tree.

![Output Classification Tree](image)

Figure 3. Output Classification Tree

(3) **Automation Test**

After data modeling, we performed automation testing with minimal human assistance on top of the model. Automation testing can increase coverage for test data and come up with more concluding test results for the selected mobile app. We used Appium as an automation tool to perform automation on the mobile app. Appium acts as a server that launches the app into the simulator or a real device and can access the elements for processing the actions triggered by the automation script which we wrote in Java. Steps to perform the automation were:

1. Install Appium server.
2. Create the automation environment for Android.
3. Create the automation environment for iOS.
4. Launch simulator/ Connect a real device.
5. Install Eclipse.
6. Create a maven project in Eclipse to write and run the automation script.

We provide the dependencies of Appium, Selenium, TestNG in the Project Object Model and then start writing the scripts. We use TestNG to run our automation tests. Soon after the execution of tests, test results are visible in the Eclipse console.

For the algorithm of the app automation, one image which is selected from the gallery of the phone is fed as an input into the target app, and the result of the execution is compared with the expected output. If the output from the target app is as expected, then the test case is displayed as passed or else failed. Also, when the app produces the output, more options, as provided by the app are taken into account. While showing the output to the user, there is an option to see more options from the suggestions coming from the app. The algorithm considers all those options as the output from the app and then decides if the test case is passed or failed.

**B. Test Result**

After applying manual testing and automation testing, we compare the coverages for both manual and automation tests. In manual testing, the coverage of the test case was limited due to timing. It was difficult to cover a larger set of data without the use of tools or scripts. On the other hand, automation testing has higher coverage because the tools and script helped us to cover more test cases. Figure 4 below shows that in automation testing we were able to cover more test sets of data than the manual testing over the same time. Approximately, in the automation testing, we were able to cover twice of what we covered in the manual testing.

![Test Coverage for Manual and Automation Test](image)

Figure 4. Test Coverage for Manual and Automation Test

The app was able to detect objects, recognize them, and classify them with its name. However, it does not tell the count or sub-classification of the food item. Moreover, testing Calorie Mama App, required a lot of time to do both manual testing and automation testing. Manual testing needs to take more time to
generate all decision tables, analyze different test causes and test manually. On the other hand, in automation testing, we spend days to get the script working correctly and program it to do the testing automatically.

The following figure shows the results of the manual testing and automation testing of the Calorie Mama APP. In manual testing, the total test food item across different cuisines was 400 items and each cuisine has 80 food items. The 132 of them were wrongly detected they were bugs in the app. This gives us a 33 failed percentage and the passing percentage is 67. The diagram below shows the failing and passing results.

In Automation testing, we tested 400 different images in different cuisines similarly. We found out that out of the 400 images, 175 failed and 225 passed. This gives us a failure percentage of 43.75 and a passing percentage of 56.25 as shown in figure 6.

Comparing the manual testing with automation testing, we can see that the errors that were found in the automation testing are higher than the errors that were found by the manual testing because the automation test allows us to test different inputs in a short time. Also, in manual testing, it is more likely to make human mistakes because doing repeated tasks over time generates more errors by humans. However, doing a repeated test using automation by writing a script and let the machine discover the error is more efficient. Therefore, automation testing discovers more errors than manual testing.

IV. CONCLUSION

To sum up, we mainly leverage two methods to test the image recognition system, namely manual testing, and automation testing. In manual testing, the test is conducted by human testers inputting the use cases one by one, and observing the results. Manual testing can be expensive and time-consuming. Moreover, it is subject to human error; therefore, it is not one hundred percent accurate. On the other hand, in automation testing, the testers use tools and scripts to help them conduct the test among the image recognition software, which can save labor and time cost, thus improving testing efficiency. It helps them find errors without the need of performing redundant tasks. However, it needs talented and experienced people to do that, which is expensive. Besides, it is difficult to automate all kinds of testing where not everything can be redundant and reusable.

ACKNOWLEDGMENT

This work is supported by the Foundation of Graduate Innovation Center in Nanjing University of Aeronautics and Astronautics (kfjj20201603).

REFERENCES


A Novel Text Classification Approach based on Meta-path Similarities and Graph Neural Networks

Huan Wang, Jiang Li*, Qing Zhou*, Liang Ge
College of Computer Science, Chongqing University, Chongqing, China
{whuan, lijfrank, tzhou, geliang}@cqu.edu.cn
*Corresponding authors: Jiang Li (lijfrank@cqu.edu.cn) and Qing Zhou (tzhou@cqu.edu.cn)

Abstract—With the rise of neural networks, studies on text classification have transitioned from traditional methods to deep learning, especially to graph neural networks on text graphs constructed from corpora. In this paper, we model the complex instances and rich interactions in text classification as a heterogeneous graph. Nevertheless, due to the overlook of indirect relations between documents, graph neural networks have not been fully exploited for the heterogeneous text graph with different types of nodes and links. Consequently, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification. Specifically, we first construct a heterogeneous text graph from corpora; we then transform the text graph into several homogeneous weighted graphs via some pre-defined meta-paths; we also propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation. Empirical results on multiple benchmark datasets have proved that our proposed method outperforms state-of-the-art graph-based methods like Text GCN.

Keywords—heterogeneous graph, text classification, natural language processing, graph neural networks, meta-path.

I. INTRODUCTION

Natural Language Processing (NLP) is a significant research direction in the field of computer science and artificial intelligence, in which text classification is one of crucial and classical tasks. The purpose of text classification is to annotate a given text sequence with one (or multiple) class label(s) describing its textual content [1]. Traditional text classification methods rely heavily on feature engineering and have stringent requirements on the input text data. Recently, neural network models have been exploited for text classification such as Convolutional Neural Networks (CNNs) [2] and Recurrent Neural Networks (RNNs) [3]. In order to increase the representation flexibility of such models, the attention mechanism has been introduced as a component of text classification model. Although these methods are effective, they cannot directly process graph-structured data, which leads to the loss of link information in a corpus.

Graph Neural Networks (GNNs), as deep learning techniques for graph-structured data, have shown superior performance and have attracted widespread attention [4]. For text classification based on GNNs, researchers need to first construct a graph from the text corpus. Zhang et al. [5] improved Defferrard et al.'s [6] work by applying word co-occurrence and document-word relations. However, this method ignores the document-document relationships and fails to capture semantic information in the heterogeneous network. The constructed text graph is usually a heterogeneous graph containing different types of vertices and links. Such text heterogeneous graphs integrate complex objects and rich semantic information, and are not fully considered in general GNNs, e.g., Graph Convolutional Network (GCN) [7] and Simplifying Graph Convolutional Network (SGC) [8] are only suitable for homogeneous graphs.

Taking into account the limitations of existing solutions, we hold the opinion that it is of critical importance to propose a method that can be used to heterogeneous text graph classification. In this work, we propose a novel text classification method based on meta-path similarities and graph neural networks, which is equipped with the following steps to effectively tackle the challenge of heterogeneous text graph classification: 1) we construct a heterogeneous text graph, which integrates rich semantic relations and structural information from the text corpus; 2) we transform the text graph into several homogeneous weighted-graphs based on some pre-defined meta-paths, where the edge weights depend on the document similarities of each meta-path; and 3) we propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation, which contains node-level and semantic-level aggregation. In the node-level aggregation, the graph convolution network is employed to integrate the neighboring document representations by weight; in the semantic level aggregation, the attention mechanism is adopted to fuse the document representations from different homogeneous graphs by weight.

Our proposed model named MPTGNN can flexibly utilize the rich interactive and semantic information in heterogeneous graph due to the consideration of meta-path. The overall model can be optimized via backpropagation in an end-to-end fashion. Our main contributions in this paper are as follows:

• We propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification, the complex heterogeneous text graph is converted into multiple homogeneous weighted graphs. In addition, the homogeneous graphs based on meta-path contain rich structural information and semantic relations.

• A Two-stage Multi-graph Information Fusion method (TMIF) is proposed for document representation, in which multiple weighted homogeneous graphs are used.

DOI reference number: 10.18293/SEKE2021-006
as inputs of GCN to obtain multiple document representations, and an attention mechanism is employed to fuse multiple document representations by weight in semantic-level aggregation.

- Results on several benchmark datasets have demonstrated that our proposed method is of effectiveness and outperforms state-of-the-art graph-based approach for text classification. It promotes the development of text classification method based on graph model.

II. RELATED WORK

A. Text Classification

Traditional text classification studies mainly focus on feature engineering and classification algorithm [9]. For feature engineering, the most commonly used methods are one-hot encoding, TF-IDF and word2vec. Some recent studies [10], [11] convert texts into graphics and extract path-based features for classification. For classification algorithm, the frequently used methods are K-Nearest Neighbor, Naive Bayes, Support Vector Machines and so on. Although these traditional techniques have succeeded in carefully edited and formal texts, they perform worse for general texts.

The research of text classification based on deep learning revolves around word embedding model and deep neural network. Several recent studies [1], [12] have shown that the success of text classification based on deep learning depends largely on the effectiveness of word embedding. Some authors aggregate unsupervised word embeddings as document embeddings [13], while others jointly learn word, document and label embeddings [14]. For deep neural networks, the most representative models are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). Kim et al. [15] uses a single-layer CNN for sentence-level classification tasks to achieve promising results. Conneau et al. [16] verify the possibility of character-level CNN to complete text classification tasks. Zhang et al. [17] use LSTM, a specific type of RNN, to learn text representation. Nevertheless, these methods rely heavily on the amount of training data and are insufficient to capture complex semantic information due to the overlook of the relations among documents or words.

Yao at al. [5] take inspiration from the recent developments of GNNs to propose a method termed Text GCN for text classification. They turn text classification problem into a node classification problem, which captures high order neighborhoods information. The work of Text GCN mainly includes two parts: 1) they regard words and documents as nodes and construct a large graph from an entire corpus; and 2) they put this graph as input into GCN to train a model. According to Text GCN, its adjacency matrix $A$ is defined as follows:

$$A_{ij} = \begin{cases} \text{PMI}(i, j) & i, j \text{ are words, PMI}(i, j) > 0 \\ \text{TF-IDF}_{ij} & i \text{ is document, } j \text{ is word} \\ 1 & i = j \\ 0 & \text{otherwise} \end{cases}$$

where $A_{ij}$ represents the weight of edge between node $i$ and node $j$. They employed point-wise mutual information (PMI) to calculate the weights between two word nodes, and treated term frequency-inverse document frequency (TF-IDF) as the weight of the edge between a document node and a word node. They actually constructed a heterogeneous graph, but simply fed it into the GCN as a homogeneous graph. Therefore, Text GCN may cause inaccurate classification accuracy due to missing rich semantic and structural information.

B. Graph Neural Networks

Recently, Graph Neural Networks (GNNs) have achieved success in processing graph-structured data, which has certified its virtue on modeling behaviors in networks [18]. Li et al. [19] presented a propagation model that incorporates gated recurrent units to propagate information across all nodes. Kipf et al. [7] proposed a spectral approach, called Graph Convolutional Network (GCN), which designs a graph neural network model via a localized first-order approximation of spectral graph convolutions. GCN is a multi-layer neural network, which directly operates on a homogeneous graph and obtains the node’s embedding vector by learning its neighborhood information. More formally, consider an undirected graph $G = (V, E)$, where $V$ and $E$ are sets of nodes and edges, respectively. Let $X \in \mathbb{R}^{n \times m}$ be a matrix containing node features $x_v \in \mathbb{R}^m$. The layer-wise propagation rule for GCN is as follows:

$$H^{(l+1)} = \sigma(\tilde{A}H^{(l)}W^{(l)})$$

where $\tilde{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is the symmetric normalized adjacency matrix, $\tilde{A} = A + I$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $W^{(l)}$ is a layer-specific trainable weight matrix, and $\sigma(\cdot)$ denotes an activation function. $H^{(l)} \in \mathbb{R}^{n \times m}$ is the hidden representation matrix for nodes in the $l$-th layer. Initially, $H^{(0)} = X$.

Wu et al. [8] presented a simple linear model, named Simplifying Graph Convolutional Network (SGC), which repeatedly eliminates the nonlinearity between GCN layers and folds the resulting function into a linear transformation. These graph neural networks can only be applied to homogeneous graphs, and cannot fully deal with heterogeneous graphs containing various types of nodes and links.

C. Meta-path

A meta-path $\mathcal{P}$ is defined as a path in the form of $O_1 \xrightarrow{R_1} O_2 \xrightarrow{R_2} \ldots \xrightarrow{R_{l-1}} O_l$ (abbreviated as $O_1 \cdot O_2 \ldots O_l$), which describes a composite relation $R = R_1 \circ R_2 \circ \ldots \circ R_{l-1}$ between the pair of types $O_1$ and $O_l$. The commuting matrix is defined by Sun et al. [20] to compute the frequencies of all the paths related to a meta-path. Given a graph $\mathcal{G} = (V, E)$, a commuting matrix $M_{\mathcal{P}}$ for a metapath $\mathcal{P} = (O_1 \cdot O_2 \ldots O_l)$ is defined as $M_{\mathcal{P}} = A_{O_1, O_2} \cdot A_{O_2, O_3} \ldots \cdot A_{O_{l-1}, O_l}$, where $A_{O_i, O_j}$ is the adjacency matrix between types $O_i$ and $O_j$, $M_{\mathcal{P}}(i, j)$ represents the number of path instances between objects $x_i$ and $y_j$.

Given a user-specified meta-path $\mathcal{P} = (O_1 \cdot O_2 \ldots O_l)$, we should calculate the similarity of a pair of objects $x \in O_1$ and
There are several straightforward similarity measures: path count, random walk or pair-wise random walk. These measures, however, are biased towards highly visible objects or highly concentrated objects, so they cannot capture equivalent semantics [20]. According to Sun et al., PathSim between two objects of the same type contains all highly concentrated objects, so they cannot capture equivalent measures, however, are biased towards highly visible objects or path count, random walk or pair-wise random walk. These perceptron for text classification.

A. Overall Framework

In order to make full use of the rich interactive information between objects in the text graph, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) to learn the structural feature representation of documents. The method proposed in this paper converts a heterogeneous text graph into multiple homogeneous graphs through meta-path, which can flexibly capture the rich structural and semantic information in Heterogeneous Information Network (HIN). The entire framework of our proposed method is shown in Figure 1. The method mainly includes four steps: 1) we construct a heterogeneous text graph from corpora, which contains various types of nodes and links; 2) based on several pre-defined meta-paths, we transform the text graph into several homogeneous weighted-graphs, where the edge weights depend on the document similarities from each meta-path; 3) we propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation; and 4) we apply a multi-layer perceptron for text classification.

B. PathSim-based Homogeneous Weighted-graphs

As proposed method by Yao et al. [5], we build a single heterogeneous text graph \( G = (V, E, A, X) \) for the corpus based on word co-occurrence and document word relations, where \( V, E, A \) and \( X \) denote node-set, edge-set, adjacency matrix and feature-set, respectively. The set of nodes \( V \) contains all documents \( D \) and unique words \( W \), i.e., \( V = D \cup W \). The set of edges \( E \) includes two major types of relations, which are word-word edges and document-word edges. \( A_{ij} \) represents the weight of node \( i \) and node \( j \) in the heterogeneous text graph.

In our work, we introduce meta-path similarities to transform a heterogeneous text graph into several homogeneous weighted-graphs. In a heterogeneous information graph, two objects can be connected through different paths. For example, in the text graph we established, two documents can be connected through the meta-path of \( \text{document} \rightarrow \text{word} \rightarrow \text{document(DWD)} \), or through the meta-path of \( \text{document} \rightarrow \text{word} \rightarrow \text{word} \rightarrow \text{document(DWWD)} \). Therefore, we can capture the relationship between documents, which is helpful to fully capture the rich semantic information in the original heterogeneous graph. We use meta-path similarities to represent the weights of document-document connections to construct a new adjacency matrix. Since the meta-paths in the text graph we established are all symmetrical, we adopt PathSim to capture the subtlety of peer-to-peer similarities. Given a symmetric meta-path \( P \), PathSim between two objects \( x_i \) and \( x_j \) from the same type can be calculated as:

\[
S_P(i,j) = \frac{2M_P(i,j)}{M_P(i,i) + M_P(j,j)}
\]

where \( M_P \) is the commuting matrix for the meta-path \( P \), \( M_P(i,i) \) and \( M_P(j,j) \) are the visibility for \( x_i \) and \( x_j \) in the network given the meta-path.

The adjacency matrix \( A \) can be divided into \( A_{DW} \), \( A_{WD} \) and \( A_{WW} \), where \( A_{DW} \) is the adjacency matrix between type \( D \) (documents) and type \( W \) (words). We define a commuting matrix \( M_P \) for each meta-path in the text graph:

\[
M_P=(DWD) = A_{DW}A_{WD}
\]
\[
M_P=(DWWD) = A_{DW}A_{WW}A_{WD}
\]

where commuting matrix \( M_P=(DWD) \) is a weight matrix, each element denotes the sum of the weights of pair-documents with the co-owned words.

For the purpose of the reduction in time and space resources, we utilize top-\( k \) similarity search for an object \( x_i \in O_1 \) to find sorted \( k \) objects in the same type, such that \( S_P(x_i, x_j) \geq S_P(x_i, \tilde{x}_j) \), for any \( \tilde{x}_j \) not in the returning list and \( x_j \) in the returning list. The top-\( k \) similarity search is shown as follows:

\[
S_P = \text{rank}_i(S_P(x_i, \cdot), k)
\]

where \( \text{rank}(S_P(x_i, \cdot), k) \) is a ranking operation, which keeps \( k \)-largest values for the object \( x_i \) in \( S_P(x_i, \cdot) \) and assigns 0 to the rest. We can convert the heterogeneous text graph to several homogeneous weighted-graphs via meta-path similarity, obtaining \( S_P=(DWD) \) and \( S_P=(DWWD) \) for all the document nodes.

C. Two-stage Multi-graph Information Fusion

We propose a Two-stage Multi-graph Information Fusion method (TMIF) for document representation, including node-level aggregation and semantic-level aggregation. The node-level aggregation integrates the influence from neighboring
document representations via graph convolutional network, while the semantic-level aggregation integrates the influence from different homogeneous graphs through the attention mechanism.

1) Node-level Aggregation: For node-level aggregation, we perform the weighted integration of neighboring document representations based on weighted graph convolutional network, which can fully extract the interactive information of objects in the text graph. Considering the node-level aggregation in our homogeneous weighted-graphs, a single-layer GCN based on different meta-paths can be described as follows:

\[ H_P = \tilde{S}_P X W_P \]  

(7)

where \( \tilde{S}_P = D^{-\frac{1}{2}} S_P D^{-\frac{1}{2}} \) is the symmetric normalized similarity matrix based on meta path \( P \), and \( W_P \) is the node-level trainable weight matrix based on the meta-path \( P \). \( X \) is the feature matrix for all the document nodes.

2) Semantic-level Aggregation: Through node-level aggregation, we can get the node embedding of each type of meta-path, which can only denote the document representation of specific semantics. In order to learn node embedding which has more rich semantic information, we use an attention mechanism to automatically learn the importance of different meta-paths and fuse the document representations by weight in the semantic-level aggregation. With the learned weights as coefficients, we can fuse all the semantic-specific node embeddings to obtain the final embedding \( Z \) as follows:

\[ Z = \sum_{P \in \Psi} \alpha_P H_P \]  

(8)

where \( \alpha_P \) represents the learned weight vector under the meta-path \( P \) for all the node embeddings, and \( \Psi \) is the set of meta-path types. \( Z \) is the final representation matrix for all the document nodes.

To learn the importance of node representation in each meta-path, we perform nonlinear transformation and employ a semantic-level attention vector \( \mu \). The importance of node representation in each meta-path is calculated as follows:

\[ e_P = \mu^T \sigma(W h_P + b) \]  

(9)

where \( W \) is the weight matrix, \( b \) is the bias vector, and \( \mu \) is the semantic-level attention vector. \( h_P \), a column from \( H^T_P \), is a learned node embedding from the node-level aggregation. For the meaningful node comparison, all the above parameters are shared for the semantic-specific node embeddings in all meta-paths.

After obtaining the importance of node representation in each meta-path, we normalize them through a softmax function. The weight of node representation in the meta-path \( P \), denoted as \( \alpha_P \), is calculated as follows:

\[ \alpha_P = \frac{\exp(e_P)}{\sum_{P \in \Psi} \exp(e_P)} \]  

(10)

where \( \Psi \) is the set of meta-path types, and \( \alpha_P \) can be interpreted as the contribution of the meta-path \( P \) for the learned node embedding.

For document classification tasks, we first feed the final representation matrix to a \( \log_{\text{softmax}} \) layer; then we exploit the negative log likelihood loss over training data with the L2-norm. The details are defined as follows:

\[ V = \log_{\text{softmax}}(Z^{(l)}) \]  

(11a)

\[ L = - \sum_{i \in D_{\text{train}}} V_{im} + \eta \| \Theta \|_2 \]  

(11b)

where \( Z^{(l)} \) denotes the \( l \)-th layer document embedding. \( D_{\text{train}} \) is the set of document node indices for training, \( V_{im} \) is the predicted value, \( \Theta \) denotes other learnable parameters in the model, and \( \eta \) is regularization factor.

IV. DATA AND EXPERIMENT

A. Datasets and Baselines

We conduct extensive experiments on 5 benchmark text datasets including MR, Ohsomed, R8, R52 and AGNews. The statistics for preprocessed datasets are summarized in Table I.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Documents</th>
<th>#Train</th>
<th>#Test</th>
<th>#Words</th>
<th>#Classes</th>
<th>Average Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR</td>
<td>10,662</td>
<td>7,106</td>
<td>3,556</td>
<td>18,764</td>
<td>2</td>
<td>20.39</td>
</tr>
<tr>
<td>Ohsomed</td>
<td>7,400</td>
<td>3,357</td>
<td>4,043</td>
<td>14,157</td>
<td>23</td>
<td>135.82</td>
</tr>
<tr>
<td>R8</td>
<td>7,674</td>
<td>5,485</td>
<td>2,189</td>
<td>7,688</td>
<td>8</td>
<td>65.72</td>
</tr>
<tr>
<td>R52</td>
<td>9,100</td>
<td>6,532</td>
<td>2,568</td>
<td>8,892</td>
<td>52</td>
<td>69.82</td>
</tr>
<tr>
<td>AGNews</td>
<td>6,000</td>
<td>4,000</td>
<td>2,000</td>
<td>9,402</td>
<td>4</td>
<td>7.62</td>
</tr>
</tbody>
</table>

- **MR**: A movie review dataset for binary sentiment classification, in which each movie review contains only one sentence [21]. The corpus has 5,331 positive and 5,331 negative reviews.
- **Ohsomed**: It is a bibliographic database of important medical literature maintained by the National Library of Medicine, which is from the MEDLINE database. We divide training set and test set according to text GCN.
- **R8 and R52**: They are two subsets of the Reuters 21578 dataset, which is a collection of documents that appeared on Reuters newswire in 1987.
- **AGNews**: We randomly selected 6,000 pieces of news from AGNews, evenly distributed into 4 classes. The ratio of training set and test set is 2:1.

In order to evaluate our method comprehensively, we compare it with the following methods. Text GCN and SGC are graph-based techniques.

- **CNN**: CNN [2] with and without pre-trained word embeddings, named CNN-rand and CNN-pretrain, respectively.
- **LSTM**: LSTM [3] with and without pre-trained word embeddings, named LSTM-rand and LSTM-pretrain, respectively.
- **fastText**: FastText [22] is a simple and efficient approach for representation learning, which treats the average of

http://www.cs.cornell.edu/people/pabo/movie-review-data/
http://dipi.unin.it/moscitti/corpora.htm
https://www.cs.umb.edu/~smimarog/extmining/datasets/
http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

567
word/n-grams embeddings as document representations, then feeds document representations into a linear classifier.

- **Text GCN**: Text GCN [5] builds a graph for the corpus based on word co-occurrence and document word relations, then applies GNN for text classification.

- **SGC**: SGC [8] is a simple linear model by simplifying the graph convolutional network. We apply SGC with the text graph established by Text GCN for text classification.

### B. Experimental Settings

In the construction of heterogeneous text graph part, we set the window size as 20. In the meta-path similarities part, we set $k = \text{rate} \times \#\text{Documents}$ for the top-$k$ similarity search, where the rate is 0.4 for MR and R52, the rate is 0.45 for Ohsumed, and the rate is 0.55 for R8 and AGNews. In the node-level aggregation part, we set the hidden dimension as 64 and the dropout rate as 0.5. In the semantic-level aggregation part, we set the dimension of the semantic-level attention vector $\mu$ to 64, and the dropout rate as 0.1. Furthermore, we set the layer number $l$ of MPTGNN as 1, set the regularization factor $\eta = 1e-5$, use Adam algorithm with learning rate 0.03 to train MPTGNN up to 500 epochs, and stop training if the validation loss does not decrease for 10 consecutive epochs. For baseline methods using pre-trained word embeddings, we use 300-dimensional Google’s word2vec word embeddings.

### C. Experimental Results and Analysis

The node classification accuracies of different methods on benchmark datasets are shown in Table II. We can seeTABLE II TEST accuracies (%) of different methods on benchmark datasets. We run each model for 10 times and record its mean ± standard deviation. The best results are in bold, and the second-best results are underlined.

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>Ohsumed</th>
<th>R8</th>
<th>R52</th>
<th>AGNews</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-rand</td>
<td>72.11 ± 0.70</td>
<td>51.85 ± 1.72</td>
<td>96.27 ± 0.24</td>
<td>92.00 ± 0.51</td>
<td>57.66 ± 0.98</td>
</tr>
<tr>
<td>CNN-pretrain</td>
<td>75.96 ± 0.32</td>
<td>68.60 ± 0.39</td>
<td>97.10 ± 0.09</td>
<td>93.57 ± 0.20</td>
<td>76.72 ± 0.65</td>
</tr>
<tr>
<td>LSTM-rand</td>
<td>70.05 ± 0.77</td>
<td>31.68 ± 1.27</td>
<td>93.77 ± 0.51</td>
<td>88.20 ± 0.60</td>
<td>60.71 ± 0.90</td>
</tr>
<tr>
<td>LSTM-pretrain</td>
<td>75.00 ± 0.50</td>
<td>45.55 ± 2.01</td>
<td>95.90 ± 0.85</td>
<td>86.62 ± 1.81</td>
<td>72.00 ± 1.19</td>
</tr>
<tr>
<td>fastText-pretrain</td>
<td>76.00 ± 1.36</td>
<td>58.16 ± 0.78</td>
<td>96.23 ± 0.31</td>
<td>90.50 ± 0.31</td>
<td>76.85 ± 0.68</td>
</tr>
<tr>
<td>fastText-pretrain-bigrams</td>
<td>73.22 ± 0.56</td>
<td>46.60 ± 0.88</td>
<td>94.98 ± 0.28</td>
<td>88.70 ± 0.79</td>
<td>62.39 ± 0.95</td>
</tr>
<tr>
<td>Text GCN</td>
<td>76.62 ± 0.13</td>
<td>68.26 ± 0.30</td>
<td>97.18 ± 0.09</td>
<td>93.68 ± 0.09</td>
<td>76.52 ± 0.14</td>
</tr>
<tr>
<td>SGC</td>
<td>76.91 ± 0.02</td>
<td>69.50 ± 0.02</td>
<td>96.39 ± 0.04</td>
<td>94.17 ± 0.05</td>
<td>70.05 ± 0.00</td>
</tr>
<tr>
<td>MPTGNN</td>
<td>77.23 ± 0.33</td>
<td>69.63 ± 0.45</td>
<td>97.13 ± 0.11</td>
<td>93.81 ± 0.29</td>
<td>77.60 ± 0.30</td>
</tr>
</tbody>
</table>

Our method outperforms all the methods on a lot of datasets, which showcases the effectiveness of our proposed method. The reasons why MPTGNN works well include that 1) we propose a method based on meta-path similarity to effectively capture the indirect relationships between documents; 2) the homogeneous weighted-graphs we constructed contains rich interactive and semantic information; 3) the top-$k$ similarity search can select the top-$k$ document nodes with the largest edge weight with the current node, which can reduce the noise; and 4) the attention mechanism can be helpful for weighted fusion of multiple document representations based on different meta-paths.

### D. Ablation Study

1) **Effect of the meta-path similarities**: To test the effectiveness of the meta-path similarities, we compare MPTGNN with its simplified version, i.e, the meta-path similarities are removed and only GNN is applied. As shown in Table III, MPTGNN performs better than GNN, demonstrating the importance of meta-path similarities for our method.

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>Ohsumed</th>
<th>R8</th>
<th>R52</th>
<th>AGNews</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN</td>
<td>76.95 ± 0.61</td>
<td>69.30 ± 0.61</td>
<td>97.01 ± 0.30</td>
<td>93.13 ± 0.73</td>
<td>77.25 ± 0.29</td>
</tr>
<tr>
<td>MPTGNN</td>
<td>77.23 ± 0.33</td>
<td>69.63 ± 0.45</td>
<td>97.13 ± 0.11</td>
<td>93.81 ± 0.29</td>
<td>77.60 ± 0.30</td>
</tr>
</tbody>
</table>

2) **Effect of the attention mechanism**: In order to test the validity of attention mechanism in the semantic-level aggregation part, we compare our model with some variants. As shown in Table IV, we compare MPTGNN with two variant models. In the semantic-level aggregation part, the concatenation and average instead of attention mechanism in MPTGNN. Text GCN based on graph neural network is a little better than CNN-pretrain on MR, R8, and R52, getting the best result for R8. The simplified neural network model SGC performs slightly better than Text GCN on MR, Ohsumed, and R52, achieving the best result for R52.

https://code.google.com/archive/p/word2vec/
ness of attention mechanism in the semantic-level aggregation part.

TABLE IV
TEST ACCURACIES (%) OF MPTGNN AND ITS VARIANTS. WE RUN EACH MODEL FOR 10 TIMES AND REPORT ITS MEAN ± STANDARD DEVIATION.

<table>
<thead>
<tr>
<th>Model</th>
<th>MR</th>
<th>Obsumed</th>
<th>R5</th>
<th>R2</th>
<th>#Docs</th>
<th>#News</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPTGNN_con</td>
<td>76.91 ± 0.23</td>
<td>70.92 ± 0.52</td>
<td>96.83 ± 0.16</td>
<td>91.58 ± 0.24</td>
<td>78.90 ± 0.10</td>
<td></td>
</tr>
<tr>
<td>MPTGNN_con</td>
<td>76.99 ± 0.28</td>
<td>68.94 ± 0.52</td>
<td>96.78 ± 0.17</td>
<td>91.51 ± 0.15</td>
<td>78.03 ± 0.33</td>
<td></td>
</tr>
<tr>
<td>MPTGNN_con</td>
<td>77.23 ± 0.33</td>
<td>66.63 ± 0.45</td>
<td>97.13 ± 0.11</td>
<td>93.81 ± 0.29</td>
<td>77.60 ± 0.30</td>
<td></td>
</tr>
</tbody>
</table>

3) Effect of the top-k similarity search: Test accuracies with different rates ($k = \text{rate} \times \#\text{Documents}$) for the top-k similarity search on MR and Obsumed are shown in Figure 2. The test accuracy affected by the rate is fluctuating, and the rate that is too close to 0 or 1 does not result in the best test accuracy. The test accuracy of MR reaches the best result when the rate is 0.4. For Obsumed, the test accuracy reaches the best result when the rate is 0.45.

V. CONCLUSION
In this paper, we propose a Meta-Path-based Text Graph Neural Network (MPTGNN) for text classification. The proposed method can effectively capture the structural and semantic information in heterogeneous text network. The meta-path similarities are regarded as the weights of document-document connections. We also apply a Two-stage Multi-graph Information Fusion method (TMIF) to learn the embedding of each document. The attention mechanism can be productive for weighted fusion of multiple document representations based on different meta-paths. Experiments on public datasets demonstrate that our proposed method can improve the performance of text classifiers. In our future work, we will consider integrating external knowledge into heterogeneous text graph, such as text topics and knowledge graphs.

ACKNOWLEDGMENT

This work was supported in part by National Natural Science Foundation Project of CQ CSTC (No. cstc2020jcyjmsxmX0554).

REFERENCES
Abstract—The interpretability of black-box text classification models has been receiving widespread attention in recent years accompanying the growing popularity of artificial intelligence. To garner user trust on the model’s decision-making process, it is imperative to provide faithful instance-wise justifications and rationalize the prediction in a human-readable way. In this paper, we address this challenge by introducing Locally Universal Rules (LURs) as model-agnostic local explanations. LURs are a subset of input words sufficient for the model to arrive at a particular prediction, even if the rest of words are perturbed slightly. We show the identification of the optimal LUR is NP-complete. Consequently, we propose a population-based algorithm LUR-Locator to perform the constrained optimization efficiently. We conduct extensive experiments to evaluate our algorithm on a cross product of well-established text classification datasets and models. The empirical results demonstrate that LUR-Locator can efficiently generate high-quality local explanations, as compared to existing explanatory methods.

Index Terms—Model-Agnostic Explanations; Local Explanations; Genetic Algorithms; Text Classification

I. INTRODUCTION

In the era of Artificial Intelligence (AI), machine learning or deep learning models have been widely deployed in a variety of real-world applications. However, due to the increasingly complex architectures, most models act as a black box without any clear explanations. The lack of interpretability can pose potential threats to the lives of individuals in today’s AI systems [1]. To establish widespread public trust on model’s behaviors, it is necessary to develop explainatory methods to provide insights into these opaque model.

One of the simplest ways to understand the decision-making process is to select a subset of input features as local explanations [2], such as a list of words for texts, a group of super-pixels for images or a set of if-else rules for tabular data, where the slight perturbations on the rest of features can hardly change the current prediction. These local explanations are presented in a feature-selective way [3], where a subset of input features responsible for the model’s prediction are selected. Compared to feature-additive local explanations [3] which compute the numerical weights for each input feature as their separate contributions to the model’s decision and provide too much redundant information, the feature-selective local explanations are very concise and straightforward. They are easy to understand even by non-expert users. For example, users may want to understand why their comments on social medias are judged as offensive sentences by the AI system, and by feature-selective local explanations they can quickly locate the key words and modify them.

In this paper, we introduce Locally Universal Rules (LUR) as model-agnostic local explanations from the perspective of feature-selection for text classification tasks, where “locally” means these explanations can only approximate the model in the neighborhood of a particular data point rather than keep faithful globally [4][5], and “universal” means the model prediction will keep the same in the perturbed distribution as long as these local explanations hold. Subsequently we propose a population-based optimization algorithm, LUR-Locator, to reduce the search space greatly when identifying the optimal LUR. We shed light on Natural Language Processing (NLP) as the driving domain here to present LURs and LUR-Locator in details, but our work can be easily extended to other domains, such as image data or tabular data.

II. RELATED WORK

Interpretable AI systems have been gaining the spotlights due to its significant value in human trust nowadays [4]. Some work perform a deep investigation on the important building blocks of model architectures such as attention [6], or on a specific family of models, such as RNN [7] and CNN [8]. Usually, model-specific explanations are more applicable for AI experts to learn the intrinsic mechanisms and debug the model [2]. The second group offer transparency on any deep learning models. They are either implemented with full access to internal parameters [9][10], or in a blackbox scenario [11]. The third group zoom in on model-agnostic explanations [4][12], and they can be used to interpret either machine learning models or deep learning models.

Local explanations are closely related to model-agnostic explanatory methods. However, when the term of local explanations is mentioned in literature, it usually reflects diverse motivations when dealing with user-specific requirements, and consequently researchers establish different but sometimes
overlapping definitions. In [13], local explanations are referred to as rationales which are short and coherent, yet sufficient pieces of text for predictions in NLP, and an encoder-generator framework is proposed to automatically generate rationales to regularize the model during training. In [4], LIME is presented to learn an interpretable model by locally approximating a particular data point. In [14], local explanations are considered as prediction interpretability (as opposed to model interpretability), and a variety of local explanatory methods are evaluated with human judgements. Anchors are formally introduced in [12] as local and sufficient conditions so that the slight perturbations on the rest of the features have negligible effects on the prediction. sufficient input subsets are presented in [2] as a minimal input pattern to make sure that the model can produce the same prediction even though all other feature values are lost. Among all the previous work, the definition of LURs is most similar to that of anchors [12]. We will demonstrate the comparison between them in experiments.

III. PROBLEM SETUP AND ANALYSIS

Firstly we present the definition of perturbation distribution:

\textbf{Definition 1 (Perturbation Distribution).} Let \( X = \{t_1, t_2, \ldots, t_n\} \) be an input document consisting of \( n \) words, and \( A = X \ast I \) be a set of words where \( I = \{0, 1\}^n \) indicates the presence or absence of each word. The perturbation distribution \( D_X(\cdot|A) \) is a conditional distribution of documents. Suppose we have a document \( X' = \{t_1', t_2', \ldots, t_n'\} \) sampled from \( D_X(\cdot|A) \). For each \( t \in X \land t \in A \), we have \( t_i = t_i' \) where \( i \) is the position of \( t \) in \( X \). For each \( t \in X \land t \notin A \), we have \( L(t_i, t_i') < \epsilon \) where \( L \) is the distance of words and \( \epsilon \) is a numerical threshold.

Refer to Figure 1 for an example. In the neighborhood of \( X_3 \), the vertical dashed line is the perturbation distribution \( D_X(\cdot|A = \{t_1\}) \) and the hollow circles are possible documents \( X' \) sampled from \( D_X(\cdot|A = \{t_1\}) \).

In perturbation-based local explanations, \( L(t_i, t_i') \) should be treated as the distance of words in a semantically meaningful space even when the words are embedded in another representation space in the model, since the perturbation distribution \( D_X(\cdot|A) \) should be interpretable to human eyes [4][12]. In reality, when drawing samples from \( D_X(\cdot|A) \), we have a set of candidate synonyms in the vicinity of \( t \). It is worth mentioning that a candidate word at the edge of the chosen area is likely to have a large semantic distance from \( t \), and consequently the sampling noise will be introduced. To improve the robustness of \( D_X(\cdot|A) \) against the sampling noise, we use the softmax function parameterized by temperature \( T \) to compute the normalized probability of words sampling:

\[
p(t_i'|t_i) = \frac{\exp(L(t_i, t_i')/T)}{\sum \exp(L(t_j, t_j')/T)} \tag{1}
\]

Based on \( D_X(\cdot|A) \), the LUR can be defined as follows:

\textbf{Definition 2 (LUR).} Let \( F : X \rightarrow \mathbb{R}^1 \) be a function representing a text classification model. Suppose \( D_A = \{X' | X' \sim D_X(\cdot|A)\} \) is a dataset approximating the true distribution of \( D_X(\cdot|A) \). The locally universal rule (LUR), denoted by \( A \), is such a set of words satisfying \( \text{Acc}(A) = \mathbb{E}_{D_A}[F(X) = F(X')] \geq \tau \), where \( \text{Acc}(A) \) denotes the performance achieved by \( A \) and \( \tau \in [0, 1] \) is a specified threshold serving as the lower bound on the expected performance.

We show the existence of the LUR as follows:

\textbf{Proposition 1.} For a document \( X \), a text classification model \( F \) and a threshold \( \tau \), at least one LUR exists.

\textbf{Proof:} Assume to the contrary that no LUR exists for some \( X, F \) and \( \tau \). Assign \( X \) to \( A \), and we have \( \text{Acc}(A) = \mathbb{E}_{D_A}[F(X) = F(X')] = \mathbb{E}_{\{X\}}[F(X) = F(X')] = 1 \geq \tau \). Therefore \( A \) is a valid LUR, and the original assumption must be false. So Proposition 1 is true.

Refer to Figure 1 for better understandings. Suppose \( \tau = 0.9 \). For \( X_1 \), since \( \text{Acc}(A = \{t_2\}) \approx 0.5 < \tau \), \( \{t_2\} \) is not a valid LUR. There only exists one LUR \( A = \{t_1\} \). Similarly, \( A = \{t_1\} \) or \( \{t_2\} \) for \( X_2 \), and \( A = \{t_1, t_2\} \) for \( X_3 \).

Subsequently, we define the optimal LUR as follows:

\textbf{Definition 3 (The Optimal LUR).} Suppose \( S : A \rightarrow \mathbb{R}^1 \) be a function representing the number of words in \( A \). For an optimal LUR \( A \), it should satisfy the following conditions: (1) \( \text{Acc}(A) \geq \tau \) (2) \( \exists A^*, \text{Acc}(A^*) \geq \tau \land S(A^*) < S(A) \).

The LUR with a shorter length is preferred as the optimal LUR for two reasons: (1) Local explanations should be concise for users to understand [2]. (2) A shorter local explanation is expected to cover more instances in realistic settings [12].

Now we present the identification of the optimal LUR as a constrained optimization problem formally:

\textbf{Definition 4 (The Identification Problem).} For an input document \( X \), the identification of the optimal LUR is to find such a minimal set of words \( A \), so that \( A = \arg \min_{A \subseteq X} S(A) \), subject to \( \text{Acc}(A) \geq \tau \).

We show this problem is computationally intractable:
Proposition 2. The identification of the optimal LUR is an NP-complete problem.

Proof: We prove that a simpler version of the original problem can be reduced to the NP-complete subset sum problem [15]. The simpler problem is \( A = \arg \max_{A \subseteq X} \text{Acc}(A) \). Let the set of non-negative values be \( \{v_1, v_2, \cdots, v_n\} \), and the target be \( K \). Let the embedding of \( t_i \in X \) be \( E(t_i) = (v_i, 0, \cdots, 0) \) and the embedding of its synonym \( t'_i \in X' \) be \( E(t'_i) = (0, 0, \cdots, 0) \). The classifier is defined as follows:

\[
F(X) = g(\sum_{i=0}^{n} \sum_{j} E(t_{i,j}))
\]

\[
g(a) = \begin{cases} 
0 & a \neq K \\
1 & a = K 
\end{cases} 
\]

where \( g(x) \) is a step function. To solve the problem, we have to find a set of words as \( A \), so that the sum of all the values in the embedding space equals to \( K \) exactly. Now the simplified problem suffices the subset sum problem. Hence, we have that the original identification problem is NP-complete.

IV. THE PROPOSED ALGORITHM

The possible search space of different combinations of words grows exponentially with the document length if we aim to identify the optimal LUR, and it is impossible to deal with the heavy computational burden in practice. Instead of an exhaustive search, we develop a heuristic algorithm, LURLocator, following the principle of genetic algorithms, to search for a near-optimal solution efficiently. The key operators include initialization, fitness, selection, crossover and mutation. The pseudo-code of LURLocator is shown in Algorithm 1, and the framework is illustrated in Figure 2.

The first step of our algorithm is to initialize an empty set \( D' \) (line 1), which is the set of instances sampled from the perturbation distribution and will be iteratively expanded later. All the LURs will be evaluated on \( D' \) to estimate the accuracy.

Line 2-3 implements the initialization operator, returning an array of chromosomes as the initial population \( P^0 \). We select one word \( t_{i \in n} \) (\% is the modular operation) as the LUR \( A_i \) in each chromosome \( P^0_i \). The chromosome is encoded as a binary vector of length \( n \), where the words in \( A_i \) take the value of 1 and others take 0.

The evolution continuous for \( k_{gen} \) generations in an iterative process to search for the near-optimal LUR (line 4).

In the fitness operator (line 5-8), we measure the quality of the population \( P^{i-1} \) in accuracy. Suppose \( D' \) has also been encoded with binary vectors, where the words in \( X \) take the value of 1 and others take 0. Firstly we count the number of \( X' \in D'_{A_j} \) in \( D' \), and at least \( B_1 \) instances are expected to estimate the accuracy of \( A_j \) (line 8). If there are not enough instances, we sample \( B_1 \) instances from the perturbation distribution \( D_X(\cdot|A_j) \) immediately to expand \( D' \) (line 6-7). The prediction results through \( F \) will be recorded, so that we do not have to perform repetitive predictions for the same instance in the subsequent iterations.

Algorithm 1 The LURLocator Algorithm

**Input:** Input document \( X = \{t_1, t_2, \cdots, t_n\} \); The function \( F \) as the text classification model; The function \( S \) as the number of words in a LUR; Batch sizes \( B_1, B_2 \); The expected performance \( \tau \); Number of generations \( k_{gen} \); Size of the population \( k_{pop} \); Number of parents \( k_{par} \); Number of mutations \( k_{mut} \)

**Output:** The near-optimal LUR \( A \).

1: \( D' = \emptyset \)
2: for \( i = 1, 2, \cdots, k_{pop} \) do
3: \( P^0_i \leftarrow \text{Encode } A_i = \{t_{i \in n}\} \subseteq X \) as a binary vector
4: for \( i = 1, 2, \cdots, k_{gen} \) do
5: for all \( A_j \in P^{i-1} \) do
6: if \( D'_{A_j}.\text{size} < B_1 \) then
7: \( D'_{A_j} \leftarrow \text{Sample } B_1 \) instances from \( D_X(\cdot|A_j) \)
8: \( \text{Acc}^c(A_j) \leftarrow \mathbb{E}_{D'_{A_j}}[F(X) = F(X')] \)
9: \( \text{Fit}^i \leftarrow \{P^{i-1}\text{size} \geq k_{par}) \text{ then} \)
10: \( \text{Fit}^i \leftarrow \text{Sort } \text{Fit}^i \text{ so that } \forall j_1 \leq j_2, S(A_{j_1}) \leq S(A_{j_2}) \)
11: \( \text{Parent}^i \leftarrow \text{Fit}^i_1, \text{Fit}^i_2, \cdots, \text{Fit}^i_{k_{par}} \)
12: \( \text{Child}^i_1 \leftarrow \text{Randomly select two individuals from Parent}^i \text{ and recombine them} \)
13: \( \text{Child}^i_2 \leftarrow \text{Randomly flip } k_{mut} \text{ genes in Child}^i_2 \)
14: \( \text{Parent}^i \leftarrow \text{Parent}^i + \text{Child}^i_2 \)
15: for all \( A \in \text{Parent}^{k_{gen}} \) do
16: \( D'_{A} \leftarrow \text{Sample } B_2 \) instances from \( D_X(\cdot|A) \) to \( D' \)
17: \( \text{Acc}(A) \leftarrow \mathbb{E}_{D'_{A}}[F(X) = F(X')] \)
20: for all \( A \in \text{Parent}^{k_{gen}} \) do
21: \( A \leftarrow \arg \min_{A \in \text{Parent}^{k_{gen}}} S(A) \), subject to \( \text{Acc}(A) \geq \tau \)
22: return \( A \)

When we attempt to select \( D'_{A_j} \) from \( D' \), we have to compare each instance in \( D' \) with the encoded chromosome \( P^{i-1}_{A_j} \) to see whether they match with each other. The comparison of 0/1 bits runs in \( O(n \times D'.\text{size}) \) time. Given a large \( n \) or \( D'.\text{size} \), the process will be time-consuming. To decrease the number of comparisons and accelerate the selection of \( D'_{A_j} \), we propose an alternative approach based on inverted index. We transpose the matrix of \( D' \), so that each row represents a word in \( X \) and each column represents an instance in \( D' \). For each word in \( A_j \), we select the corresponding rows in the transposed matrix and multiply these 0/1 rows in an element-wise manner. In the resulting vector of length \( D'.\text{size} \), the elements are equal to 1 only if all of its multiplicators are equal to 1. Refer to the second subgraph of the second row in Figure 2 for an example, where \( A_j = \{t_4, t_5\} \). When we compute the element-wise multiplication of \( t_4 = (0, 1, 0, \cdots) \)
1. Initialization

\[
\begin{align*}
P = & \{ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ \vdots \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \} \\
A = & \{ \begin{bmatrix} t_1 \\ 0 \\ \vdots \\ t_5 \end{bmatrix} \}
\end{align*}
\]

2. Fitness

\[
\begin{align*}
P & = \{ \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \} \\
P & = \{ \begin{bmatrix} 0 & 1 & \cdots & 0 \end{bmatrix} \}
\end{align*}
\]

3. Selection

\[
\begin{align*}
Parent' = & \{ \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \\
& \begin{bmatrix} 0 & 1 & \cdots & 0 \end{bmatrix} \\
\end{align*}
\]

4. Crossover

\[
\begin{align*}
Child' = & \{ \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \\
& \begin{bmatrix} 0 & 1 & \cdots & 0 \end{bmatrix} \\
\end{align*}
\]

5. Mutation

\[
\begin{align*}
Child = & \{ \begin{bmatrix} 0 & 0 & \cdots & 1 \end{bmatrix} \\
& \begin{bmatrix} 0 & 1 & \cdots & 0 \end{bmatrix} \\
\end{align*}
\]

\[
\begin{align*}
D & = \{ \begin{bmatrix} D_1, \cdots, D_{n/k} \end{bmatrix} \} \\
\end{align*}
\]

\[
\begin{align*}
& \text{Select } D_i \text{ From } D' \\
& \text{Encode } D_i \\
& \text{Perturbed Distribution } P = \{ \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix} \} \\
& \text{Select } D_i \text{ From } D' \\
& \text{Encode } D_i \\
& \text{Perturbed Distribution } P = \{ \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \end{bmatrix} \}
\end{align*}
\]

Fig. 2. The framework of LURLocator

and \( t_5 = (0, 1, \ldots, 0) \) in the transposed matrix and get the result of \( (0, 1, \ldots, 0) \), only the second element (indicating \( D_2' \)) is equal to 1, meaning that \( D_2' \) matches \( A_j = \{ t_4, t_5 \} \) and belongs to \( D_{A_j} \). In other words, the index of the element equal to 1 in the resulting vector is the position of the instance in \( D' \) that belongs to \( D_{A_j} \). This alternative approach runs in \( O(S(A_j) \times D'.size) \) time and achieves a speed-up of \( (n/S(A_j)) - 1 \) compared to the original implementation. The gains in efficiency is especially significant when \( S(A_j) \ll n \).

We implement the selection operator in line 9-15. We filter out the chromosomes whose LURs reach the performance threshold of \( \tau \), denoted as \( Fit_t \) (line 9). If the size of \( Fit_t \) is larger than \( k_{par} \), it means that we already have enough chromosomes as parents, from which we can select the ones encoded with the \( k_{par} \) shortest LURs as \( Parent' \) for future breeding (line 10-12). Otherwise, we sort all the chromosomes in \( P^{t-1} \) according to \( Acc'^t \), and retrieve the ones achieving the \( k_{par} \) best accuracies (line 13-15). This branch is to make sure that if there are not enough qualified chromosomes achieving the expected fitness, we can still preferentially select more fit individuals with as high accuracy as possible.

In crossover (line 17), a set of children will be generated by randomly choosing pairs of parents and performing crossover operations where the cut point is at the center of each parent. In this way, the new offspring has the genes from two different parents whose characteristics can be partly inherited. In the stage of mutation (line 18), \( k_{mut} \) genes in the children in the representation of binary vectors will be flipped (1 to 0 or 0 to 1). Both crossover and mutation operators ensure the diversity of individuals in the subsequent new population, so that we can explore more candidate solutions.

Both \( Parent' \) and \( Child' \) will be added together as the \( i \)th generation of the population (line 19), for the purpose that both the qualified solutions (\( Parent' \)) and the possible candidate solutions (\( Child' \)) can be safely preserved.

After \( k_{gen} \) generations, we select the optimal LUR from \( Parent^{k_{gen}} \) with the shortest length and the expected accuracy (line 20-24). To avoid over-fitting on \( D' \), we re-sample \( B_2 \) instances as \( D'_{A_i} \) to evaluate each possible LUR. If no LUR satisfies the expected \( \tau \), we return \( A = X \) as the near-optimal LUR according to Proposition 1. If more than one LUR meets the criteria, we return the LUR with the highest accuracy.

V. EXPERIMENTS

A. Experimental Setup

Datasets: We prepare four different benchmarking datasets widely used in the task of text classification, including the sentence polarity dataset from Rotten Tomatoes web pages (RT) [16], AG’s News (AG), DBPedia (DBP) and Yahoo Answers (Yah) [17].

Machine Learning Models: We train three different machine learning models, including logistic regression (LR), multinomial naive bayes (NB) and support vector machine (SVM). We use Anchor [12] as the baseline to generate the local explanations here. As we will show later, Anchor is very inefficient when dealing with long documents, so we perform the comparisons on RT whose documents are shorter.

Deep Learning Models: We choose two popular text classification models, including FastText and TextCNN, as the target deep learning models. Later, we will evaluate some gradient-based explanatory methods specially designed for neural networks, so we distinguish the experiments on deep learning models from the traditional machine learning models.

Algorithm Details: We set \( B_1 = 100, B_2 = 500, k_{gen} = 10, k_{pop} = 10 * n, k_{par} = n, k_{mut} = 1 \), where \( n \) is the length of the document. For Yah dataset, we set \( k_{gen} = 20 \) since it consists of longer documents. We present some practitioners’ guides for the fine-tuning of parameters: \( B_1 \) and \( B_2 \) are closely related to the trade-off between the algorithm’s efficiency and the LUR’s accuracy estimation. A larger \( k_{gen} \) improves the possibility of identifying the near-optimal LUR whose natural length is long; A larger \( k_{pop} \) contributes to finding the LUR as short as possible; The size of \( k_{par} \) should be at least \( n \) to ensure a good start point; We suggest setting \( k_{mut} = 1 \) to ensure a steady evolution. We use the 100 nearest words in the pre-trained embedding space from the vocabulary of each dataset to generate the perturbation distribution instead of setting a fixed \( \epsilon \). The performance threshold \( \tau \) is set to 0.9.

B. Evaluating LURs in Machine Learning

On the test set of RT, we generate LURs with our proposed LURLocator algorithm. We also generate anchors [12] with their original implementations, including the greedy-search
algorithm and the beam-search algorithm. For each local explanation \( A \), we evaluate it on \( D'_A \) which consists of 10000 test instances sampled from \( D_X(\cdot|A) \). The average test accuracy has been reported in Table I. Besides, we also report the average length of local explanations as well as the average time for the generation of one local explanation. The time cost of searching synonyms in the vocabulary has been excluded from the reported time, because synonyms can be identified in advance and saved in a hash table.

It can be clearly seen that our algorithm produces reasonable accuracy gains compared to the two baselines. The average length is comparable for LURs and anchors while significantly shorter than the average document length, so both local explanations achieve a similar level of conciseness and coverage in practical use. In terms of the running time, LURLocator leads to better efficiency compared to the two baselines.

Since genetic algorithms are stochastic and it is impossible to report the average-case complexity for LURLocator, we analyze it in the worst case. The time overhead mainly depends on model predictions. If we need to sample a new batch of instances every time, model predictions run in \( O(k_{gen} \times k_{pop} \times B_1 + k_{par} \times B_2) \) time. Given our suggested parameters, \( O \) scales linearly with the document length \( n \).

To further investigate the relationship between the document length and the required time, we report the distribution of time cost in Figure 3. For anchors, we only report the greedy-search algorithm since the beam-search algorithm requires much more time. Along the vertical direction, most green markers are above the red markers, and the distribution of green markers is more discrete. We attribute this observation to the fact that LURLocator is more efficient and stable when dealing with a fixed-length document. Along the horizontal direction, while the time cost of both algorithms keeps increasing with document length, our algorithm shows slower growth with a higher Pearson’s correlation coefficient. The results demonstrate that our algorithm has better scalability when dealing with longer documents.

We display some examples in Table II. Interestingly, the selected words from LURs and anchors are often overlapped with each other. A possible explanation is that these words contain the most important information for the current prediction. Besides, the local explanations across different models vary a lot. In fact, local explanations only reflect the model’s decision-making process rather than human reasoning, and different models can have their own views on the decision boundary, which results in different local explanations for the same instance. It is worth mentioning that the examples themselves can not indicate which explanatory method is more faithful, since the objective judgments does not necessarily align with the true model views.

C. Evaluating LURs in Deep Learning

We compare LURLocator with one perturbation-based explanatory method LIME [4] and three feature-additive methods popular in deep learning, including sensitivity analysis (SA) [9], gradients \( \times \) inputs (GI) [10] and leave-one-out (LOO) [11]. Since the last three methods assign importance scores to every input features, we select the most salient words

\[
\text{TABLE I}
\]

<table>
<thead>
<tr>
<th>Model</th>
<th>LUR</th>
<th>Anchor (Greedy Search)</th>
<th>Anchor (Beam Search)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accuracy</td>
<td>Length</td>
<td>Time</td>
</tr>
<tr>
<td>LR</td>
<td>93.31%</td>
<td>3.15</td>
<td>2.02s</td>
</tr>
<tr>
<td>NB</td>
<td>94.02%</td>
<td>3.46</td>
<td>2.03s</td>
</tr>
<tr>
<td>SVM</td>
<td>92.90%</td>
<td>3.63</td>
<td>1.99s</td>
</tr>
</tbody>
</table>

\[
\text{TABLE II}
\]

<table>
<thead>
<tr>
<th>Two generations within one family test boundaries in this intelligent and restrained coming-of-age drama. The beautiful, unusual music is this film’s chief draw, but its dreaminess may lull you to sleep.</th>
<th>LR</th>
<th>intelligent</th>
<th>family, intelligent boundaries</th>
<th>intelligent, age boundaries within, boundaries, drama within, boundaries, intelligent</th>
</tr>
</thead>
<tbody>
<tr>
<td>beautiful beautiful, draw beautiful, films beautiful, may</td>
<td>NB</td>
<td>chief</td>
<td>unusual, chief beautiful, chief</td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>beautiful</td>
<td>family, intelligent boundaries</td>
<td>intelligent, age boundaries within, boundaries, intelligent</td>
<td></td>
</tr>
<tr>
<td>SVM</td>
<td>beautiful</td>
<td>beautiful, music, draw beautiful, draw</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 3. The required time for the generation of local explanations on the test set of RT. The model is LR here. Each scatter point represents a test sample. The horizontal axis represents the document length, and the horizontal axis represents the time cost. (LURLocator: Pearson \( \rho = 0.797 \), p-value=1.27e-23; Greedy Search: Pearson \( \rho = 0.589 \), p-value=1.52e-98)
as sufficient conditions whose length is the same as LUR. We also set a Rand baseline to randomly select words as local explanations. Table III reports the average test accuracy and the average length of LURs.

We can observe that the average length of LURs on Yah is longer than other datasets because the average document length on Yah is much longer. In fact, if we increase \( k_{gen} \) or \( k_{pop} \), the search space of the optimal LUR will be expanded, and consequently a shorter average length can be achieved. Another interesting observation is that the average length of LURs on RT here is significantly shorter than the results in Table I. A possible reason is that the deep learning models based on neural networks are more robust to perturbations than the traditional machine learning models, so a shorter LUR is enough to achieve a high accuracy.

In terms of test accuracy, LURLocator always outperforms alternative solutions by a large margin. The results can be further improved if we increase the threshold \( \tau \). The Rand baseline achieves a remarkable result (over 90%) on DBP. The intuitive explanation is that the decision boundary to segment the data points on DBP is extremely robust (Refer to Figure 1 where \( X_2 \) safely locates inside the decision boundary), so that slight perturbations can hardly change the prediction. In fact, the accuracy achieved by Rand will be decreased if we increase the number of synonyms in the perturbation distribution intentionally. From the results, it can be concluded that LURs excel at providing sufficient justifications, while existing feature-additive explanatory methods have trouble selecting such a set of words.

**D. Conclusions**

In this paper, we shed light on the problem of model interpretability for text classification and introduce locally universal rules (LURs), which are a minimal set of input features sufficient to rationalize the instance-wise predictions. We propose LURLocator based on genetic algorithms to identify the optimal LUR. Extensive experiments are performed on a variety of models and datasets to evaluate the proposed algorithm as well as the generated LURs. The results show that our algorithm leads to better performance.

**REFERENCES**


**TABLE III**

Comparisons between LURs and other local explanations in deep learning.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Model</th>
<th>Length</th>
<th>LURLocator</th>
<th>LIME</th>
<th>Rand</th>
<th>SA</th>
<th>GI</th>
<th>LOO</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT.</td>
<td>FastText</td>
<td>1.79</td>
<td>99.04%</td>
<td>44.64%</td>
<td>44.92%</td>
<td>45.21%</td>
<td>45.50%</td>
<td>45.21%</td>
</tr>
<tr>
<td></td>
<td>TextCNN</td>
<td>1.71</td>
<td>98.47%</td>
<td>58.62%</td>
<td>54.69%</td>
<td>60.34%</td>
<td>64.94%</td>
<td>64.37%</td>
</tr>
<tr>
<td>AG.</td>
<td>FastText</td>
<td>2.42</td>
<td>95.48%</td>
<td>58.65%</td>
<td>57.13%</td>
<td>57.54%</td>
<td>62.31%</td>
<td>63.29%</td>
</tr>
<tr>
<td></td>
<td>TextCNN</td>
<td>2.29</td>
<td>96.52%</td>
<td>75.43%</td>
<td>71.40%</td>
<td>76.82%</td>
<td>78.60%</td>
<td>77.95%</td>
</tr>
<tr>
<td>DBP.</td>
<td>FastText</td>
<td>2.24</td>
<td>97.47%</td>
<td>92.89%</td>
<td>92.38%</td>
<td>92.71%</td>
<td>94.29%</td>
<td>93.92%</td>
</tr>
<tr>
<td></td>
<td>TextCNN</td>
<td>1.96</td>
<td>97.05%</td>
<td>92.07%</td>
<td>91.19%</td>
<td>92.64%</td>
<td>93.67%</td>
<td>93.89%</td>
</tr>
<tr>
<td>Yah.</td>
<td>FastText</td>
<td>6.88</td>
<td>93.64%</td>
<td>40.17%</td>
<td>38.90%</td>
<td>39.17%</td>
<td>41.83%</td>
<td>40.70%</td>
</tr>
<tr>
<td></td>
<td>TextCNN</td>
<td>7.95</td>
<td>93.77%</td>
<td>75.86%</td>
<td>71.25%</td>
<td>78.89%</td>
<td>80.87%</td>
<td>79.18%</td>
</tr>
</tbody>
</table>
Towards a Better Understanding of Gradient-Based Explanatory Methods in NLP

Qingfeng Du  
School of Software Engineering  
Tongji University  
Shanghai, China  
du_cloud@tongji.edu.cn

Jincheng Xu  
School of Software Engineering  
Tongji University  
Shanghai, China  
xujincheng@tongji.edu.cn

Abstract—To grasp what makes the deep learning models arrive at a particular prediction, gradient-based explanatory methods have been widely used in Natural Language Processing (NLP) recently. While the saliency maps of images can be computed directly in the pixel-level input space, the continuous gradient vector for words has to be reduced to a single value to indicate the word-level importance, and existing methods such as Sensitivity Analysis (SA) and Gradient \times Input (GI) are either tricky or short of a deep investigation. In this paper, we review the family of gradient-based explanatory methods and discuss their practical implications. Specially, we propose the signed version of GI, namely SignedGI, while some previous work may have misunderstandings on its signedness. We also show the weakness of SA-based methods. We conduct extensive experiments to evaluate these explanatory methods both qualitatively and quantitatively.

Index Terms—Gradient-based Explanatory Methods; Sensitivity Analysis; Gradient \times Input; Text Classification

I. INTRODUCTION

In the era of Artificial Intelligence (AI), deep learning models have been widely deployed in a variety of applications in Natural Language Processing (NLP), but are often criticized for the inability to explain their decisions. To afford transparency on the nested non-linear structure of the black box and shed light on interpretable AI models, a plethora of explanatory methods have been developed in literature [1] nowadays. Among existing work, gradient-based methods [2][3][4] have been gaining the spotlight recently because they can be easily used in any off-the-shelf neural networks.

It is straightforward to compute the pixel-level gradient in images [5][6], indicating how much the pixel contributes to the final prediction. However, things are different in NLP. Words are usually embedded in a continuous space, and a scalar value rather than a vector of gradients has to be derived for the word-level importance score. Consequently, many variations of gradient-based explanatory methods have been proposed in NLP to compute the scalar value, such as the sum of gradients in raw values [7], the \(L_1\) norm [3], the \(L_2\) norm [8][9][10], or the dot product between the vector of gradients and the word embedding itself [2][11][4]. For brevity, we refer to the first three variations as SA-based (Sensitivity Analysis [6]) methods, and the last one as GI (Gradient \times Input [4]).

Even though we have a sophisticated theory for gradient-based explanatory methods in images, our current understanding on them in NLP is still rudimentary. On the one hand, the SA-based methods are very tricky. The gradient measures the local effect of a particular dimension in the word vector and it does not hold water to sum up the effects along the vector as the word-level importance score, since the true changes in the embedding space are discrete rather than continuous when a word is removed or replaced. On the other hand, the correctness of GI has not been strictly proved yet, especially its signedness (A word-level importance score is signed if it can distinguish between positive and negative impacts).

In this paper, we shed light on the aforementioned gradient-based explanatory methods. We propose the signed version of GI, namely SignedGI, based on the chain rule and the backpropagation algorithm [12]. The SignedGI score is the opposite of the dot product, whereas some previous work [2][4] neglect the signedness. Besides, we show the weakness of SA-based method. We conduct extensive experiments to evaluate these explanatory methods both qualitatively and quantitatively.

II. RELATED WORKS

There has been a remarkable series of work for explainable artificial intelligence in NLP [1]. In [13], Leave-One-Out (LOO) estimates the word importance by observing the change of the log-likelihood when a particular word is removed. It has been widely used as a black-box explanatory method in NLP [7][4]. In the white-box settings, gradient-based explanatory methods have attracted great interest. As described previously, we mainly consider SA-based methods [3] and GI [2] in this paper. They aim to compute the gradient w.r.t. the word to indicate the word-level importance score. There is another popular gradient-based explanatory method named Integrated Gradients [14], which integrates over all gradients on a linear interpolation between the original input and the baseline input masked with zeros. However, the \(L_p\) norm in SA-based methods or the dot product in GI are still a prerequisite for the use of integrated gradients in NLP [15][16], so we exclude it from the scope of our work.

This work was supported by National Key R&D Program of China (Grant No. 2020YFB2103300)
DOI reference number: 10.18293/SEKE2021-045
The weakness of SA-based methods (in the $L_2$ norm) has already been noticed in the experiments of some previous work [8][15]. They attribute this observation to the fact that the $L_2$ norm can only measure the word importance with the inability to distinguish between positive and negative impacts. We will address the deeper cause in the section below, that the overall impact on the loss is uncertain when we mask the word identified by SA-based methods with all-zero paddings. Nonetheless, we are interested in the gradient of the value “1” in the one-hot vector, which indicates how much the existence of the word $w_i$ locally affects the network output.

We show how to compute the word-level gradient in $I_x$ now. For brevity, we assume that $I_x$ is a document consisting of only one word (or $m=1$), and the one-hot representation is $I_x = (t_1, \ldots, t_v) \in \mathbb{R}^{1 \times v}$ where $t_i = 0$ or 1. Assume WLOG that $t_1 = 1$ and $t_2$ to $t_v$ are all zeros. In other words, we assume that the only one word in $I_x$ corresponds to the first word in the vocabulary. Then, the output of the embedding layer $O_E$ in a neural network can be computed as follows, where $O_E = (e_1, \ldots, e_n) \in \mathbb{R}^{1 \times n}$ is the embedding for the only one word in $I_x$:

$$O_E = I_x \times E$$  \hspace{1cm} (4)

Let $l_y : x \rightarrow \mathbb{R}^1$ be the loss function w.r.t. the legitimate label $y$. We compute the word-level gradient in $I_x$ using the chain rule and the back-propagation algorithm [12]:

$$\frac{\partial l_y}{\partial I_x} = \frac{\partial l_y}{\partial O_E} \frac{\partial O_E}{\partial I_x}$$
$$= (\frac{\partial l_y}{\partial e_1}, \ldots, \frac{\partial l_y}{\partial e_n}) (\frac{\partial O_E}{\partial t_1}, \ldots, \frac{\partial O_E}{\partial t_v})$$
$$= (\frac{\partial l_y}{\partial e_1}, \ldots, \frac{\partial l_y}{\partial e_n}) \begin{bmatrix}
\frac{\partial e_1}{\partial t_1} & \cdots & \frac{\partial e_1}{\partial t_{v-1}} & \frac{\partial e_1}{\partial t_v} \\
\vdots & \ddots & \vdots & \vdots \\
\frac{\partial e_n}{\partial t_1} & \cdots & \frac{\partial e_n}{\partial t_{v-1}} & \frac{\partial e_n}{\partial t_v}
\end{bmatrix}$$
$$= (\frac{\partial l_y}{\partial e_1}, \ldots, \frac{\partial l_y}{\partial e_n}) \begin{bmatrix}
e_1 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots \\
e_n & \cdots & 0 & 0
\end{bmatrix}$$
$$= (\sum_{i=1}^{n} \frac{\partial l_y}{\partial e_i} * e_i, \cdots, 0, 0) \in \mathbb{R}^{1 \times n}$$  \hspace{1cm} (5)

The word-level gradient of $t_1$ in $I_x$ is exactly the dot product between the word embedding and the gradient w.r.t. the embedding itself. The proof here can be easily extended to the case where $I_x$ consists of more than one word.

Let the gradient of the word $w$ be $g_w$. The SignedGI scores are defined as the opposite of $g_w$, or more formally:

$$s_w = -g_w = -\sum_{i=1}^{n} \frac{\partial l_y}{\partial e_i} * e_i$$  \hspace{1cm} (6)

Now we explain the reasons. According to the algorithm of gradient descent, the parameters should move in the direction of steepest descent as defined by the negative of the gradient to minimize the loss. Here, the parameter $t_1$ can only take the value of 1 or 0, indicating the existence or non-existence of the word $w$ respectively. Hence, in the case of $g_w > 0$, the change of $t_1$ from 1 to 0 follows the gradient descent direction.
and thus decreases the loss \( l_x \). In other words, the removal of the word \( w \) has a positive impact on the performance, or, the existence of the word \( w \) has a negative impact on the performance when \( g_w > 0 \). Similarly, the existence of the word \( w \) has a positive impact on the performance when \( g_w < 0 \). By convention, the word-level importance score should be positive if the word contributes to the current prediction, so we arrive at the final representation for SignedGI as in Equation 6. Note that the signedness of GI has been neglected in its original publication [2], and been misunderstood in the subsequent work [4].

We can also analyze the SA-based methods in a similar way. Let us take \(|SA|_2\) as an example. The removal of a word identified by \(|SA|_2\) can be interpreted as masking its word embedding with all-zero paddings. In the original embedding space, the word is represented as a \( n \)-dimensional vector, and the values in various dimensions can be larger than 0 or smaller than 0. The mask of 0 will make them either follow or go against the gradient descent direction. As a result, the overall impact on the loss becomes uncertain, which leads to the weakness of SA-based methods. In summary, SA-based methods compute the gradients in the embedding level and the sum of them to indicate the word importance is inaccurate. In contrast, SignedGI computes the gradient in the word level directly, so it should be more faithful than SA-based methods.

IV. EXPERIMENTS

A. Preliminaries

Datasets We use two publicly available text classification datasets: (1) AG’s News: A topic classification dataset consisting of four categories, including World, Sports, Business, and Sci/Tech. (2) Internet Movie Database (IMDB): A binary sentiment analysis dataset on movie reviews.

Models We consider three popular text classification models, including a linear classifier FastText [18], a convolutional neural network TextCNN [19] and a bi-directional recurrent neural network BiLSTM.

Baselines Apart from the gradient-based explanatory methods as mentioned previously, we introduce two more baselines, namely Random (RD) and Leave-One-Out (LOO) [13]. The first baseline simply generates a random permutation of words to simulate the decreasing order of word importance. It can be considered as a very uninformative approach. The second baseline estimates the importance scores by erasing each word from the input and tracking the effect. The variations of LOO can be found in [13][20]. In our implementation, we compute the difference in loss:

\[
s_w = l_y(x) - l_y(x_{|w=0})
\]  

(7)

where \( l_y(x) \) is the original loss and \( l_y(x_{|w=0}) \) is the loss when masking the word embedding of \( w \) with all-zero paddings. LOO is very similar to the perturbation experiment itself (which will be introduced later). Similar baselines have also been set up in [7][4]. With the possible upper bound and lower bound on the explanatory ability, we can show the results of gradient-based methods in a more intuitive way.

The word “possible” means LOO may not produce the best explanatory ability among existing methods, but it is faithful enough. Taking it as an upper bound is helpful for us to see the difference between the results of gradient-based methods and faithful explanations. So it is with RD.

Metrics In order to evaluate the explanatory ability of different methods, existing work usually perform the perturbation-based experiment [7][4], which perturbs the original input in a word level (e.g., the mask of zero paddings, or the deletion operation), and subsequently measures the changes on the performance (e.g., the changes on accuracies, probabilities, or losses). The word importance increases monotonically with the change. Based on this observation, an objective quality measure, AOPC, is proposed in [6] to evaluate ordered collections of features quantitatively. While originally designed for images, AOPC can be easily extended to NLP [7]:

\[
AOPC = 1 - \frac{1}{K+1} \left( \sum_{k=0}^{K} f_y(x^{(0)}) - f_y(x^{(k)}) \right)_{avg}
\]

(8)

where \( x^{(k)} \) is the perturbed input with the most important \( k \) words masked with zero paddings, \( f_y(x) \) is the probability of the legitimate label \( y \), \( K \) is the cut-off point w.r.t. the top-\( K \) important words, and \( \langle \cdot \rangle_{avg} \) represents the average over all the documents. The perturbation of the most important words implies a steep decreases of \( f_y(x) \), so the method with the better explanatory ability has a larger AOPC.

In fact, AOPC values measure the absolute word importance. We can also describe the word importance in a relative way. We take LOO as a well-established benchmark because it provides the possible upper bound on the explanatory ability, and we report the Pearson correlation coefficient between the results of LOO and gradient-based methods.

\[
\rho = \frac{\sum_{i=1}^{m} (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_{i=1}^{m} (a_i - \bar{a})^2} \sqrt{\sum_{i=1}^{m} (b_i - \bar{b})^2}}
\]  

(9)

where \( a = (a_1, a_2, \ldots, a_m) \) and \( b = (b_1, b_2, \ldots, b_m) \) are the score vectors. \( \bar{a} \) and \( \bar{b} \) denote the average operation. If \( \rho \) is close to 1, there is a strong positive linear association between \( a \) and \( b \), indicating that the estimated word importance in \( b \) is as faithful as \( a \) and vice versa. On the contrary, if \( \rho \) is close to \(-1\), the relationship is strongly negative.

Others When computing the word importance with explanatory methods, existing work use either the predicted class [7] or the legitimate class [4] as the target class. Since we already have the ground-truth labels, we use the latter approach.

B. Quantitative Comparison of Explanatory Methods

We quantitatively compare the explanatory methods and compute their AOPC values varying the cut-off point \( K \) from 0 to the maximum document length. Fig. 1 illustrates the results.

Generally speaking, all the SA-based methods suffer performance decline definitely compared to SignedGI. The curves of \(|SA|_1\), \(|SA|_2\) and \(|SA|_{\infty}\) are almost overlapped and indicate the similar explanatory abilities. This is mainly because the
LOO and SignedGI always demonstrate the larger AOPC values. In other words, they better identify the important words. Even though we assume that LOO provides the upper bound on the explanatory ability in our perturbation-based experiments, SignedGI outperforms LOO in TextCNN interestingly. We attribute this observation to the weakness of LOO, that it computes the contribution of words independently without considering their mutual effects. Hence, it might result in a sub-optimal explanation [7]. In most cases, the AOPC values of SignedGI and LOO are generally comparable, except for IMDB-BiLSTM where LOO surpasses SignedGI by a large margin. A possible reason is that the long document length of IMDB causes the vanishing gradient problem in the recurrent structure, which exerts a negative impact on the SignedGI performance. Note that a deep investigation into the performance difference between LOO and SignedGI is not the

Fig. 1. Comparison of the considered explanatory methods in terms of AOPC values. The horizontal axis represents the cut-off point $K$, and the vertical axis represents the AOPC value. For each $K$, a larger AOPC value indicates the better explanatory ability of the top-$K$ important words.

Fig. 2. Histogram of Pearson correlation coefficient $\rho$ between LOO and gradient-based methods. The horizontal axis represents $\rho$ and the vertical axis represents the distribution. $|SA|_1$ is selected to represent SA-based methods since the results of others are similar. The fraction of instances whose correlation is statistically significant (p-value $< 0.05$) has been reported in the top-left corner. Note that the p-value here largely depends on the document length, so the correlation is prone to be weak.
(a) Visualization of gradient information. Each row represents the continuous word representation, where each cell is the gradient of a particular dimension in the embedding space.

(b) Visualization of instance-wise explanations. Each row displays the word-level importance scores for the instance, and the used explanatory method is indicated on the left.

Fig. 3. Qualitative comparison of explanatory methods. The instance is selected from the test set of IMDB and the target class is “positive”. The red color indicates a positive score, and the blue color indicates a negative score. The maximum value of IMDB-BiLSTM has been reduced from 5.0 to 1.0 for better visualizations, otherwise the negative color bar will be overwhelmed.

C. Correlation Between LOO and Gradient-based Methods

The full distributions of the Pearson correlation coefficient \( \rho \) have been illustrated in Fig. 2. The general observation from the figures is that SignedGI does tend to have a strong positive association with LOO, and a statistically significant correlation can be consistently established. The association on IMDB-BiLSTM seems to be less strong, which is consistent with the results in Fig. 1 that the difference between SignedGI and LOO is a bit pronounced on IMDB-BiLSTM. On the other hand, the centrality of densities for \( |S_A| \) lies in the range of 0.2~0.5 on the dataset of AG, which shows a very weak positive association with LOO. On the dataset of IMDB, the centrality hovers around 0.0, indicating almost no association. The results here further support the fact that SignedGI shows better explanatory ability compared to SA-based methods.

D. Qualitative Comparison of Explanatory Methods

In this section, we illustrate the gradient information in the embedding space in Fig. 3 (a), and we visualize the instance-wise explanations in Fig. 3 (b).

Now we take a look at the results of IMDB-FastText in Fig. 3 (a). Interestingly, the gradients keep the same in each dimension, except for the word “is”, whose gradient values are exactly twice as much as other words. In fact, in the architecture of FastText, the word embeddings are averaged directly. As a result, the gradient value of a specific dimension
in the embedding space is always proportional to the word frequency in the input document. As we can see, FastText is a pretty compelling counterexample to the effectiveness of SA-based methods. The words of the same frequency will always be assigned with the same SA-based scores, but they are embedded in various continuous representations and go through the same linear layer, meaning that their contributions to the final prediction are different in reality.

In Fig. 3 (a), the results do not have a clear focus in the heatmap of IMDB-TextCNN. In IMDB-BiLSTM, the words “bad”, “home” and “video” stands out, but the target label is “positive” and the model attaches almost zero emphasis on the positive sentiment word “great”. Note that the explanatory methods only reflect the model’s own “view” on the model prediction rather than human reasoning, so it is possible that “great” does not play an important role in the binary sentiment analysis task. However, as we will show later, “great” does have a strong positive impact in this model, and the heatmap here indeed fails to capture the relevant information.

In Fig. 3 (b), let us focus on the results of LOO firstly. It can be clearly seen that “great” has a large positive score across all three models, meaning that “great” contributes a lot to the target label. On the contrary, “bad” is always assigned with a large negative score, indicating that it has a negative impact to the current prediction. The results agree with human observations, that “great” has a positive impact and “bad” has a negative impact on the prediction of positive sentiments. Not surprisingly, the results of SignedGI are very close to LOO, except for the case on BiLSTM where SignedGI misses the word “great”, which also agrees with previous results that SignedGI works less well on BiLSTM. Nevertheless, SignedGI still filters out the important words in the qualitative experiments and provides the reasonable signed explanations. On the other hand, the results of SA-based methods are less focused. Sometimes they cannot select the important words correctly, or cannot distinguish between negative and positive impacts.

V. CONCLUSION

Gradient-based explanatory methods have been widely used in NLP nowadays. In this paper, we review existing methods and discuss their practical implications. We propose the signed version of GI, namely SignedGI, and show the weakness of SA-based methods. We conduct comprehensive experiments to evaluate different methods, and the empirical results demonstrate that SignedGI significantly outperforms SA-based methods in explanatory ability. We hope our work helps researchers to obtain the more accurate instance-wise explanations via gradient-based explanatory methods in NLP.

REFERENCES

Relation Extraction Model Based on Keywords Attention

Yu Chen  
School of Computer Science  
Hubei University of Technology  
Hubei, China  
1148848330@qq.com  

Jianxia Chen  
School of Computer Science  
Hubei University of Technology  
Hubei, China  
chenjianxiawh@gmail.com  

Chang Liu  
School of Computer Science  
Hubei University of Technology  
Hubei, China  
1323821376@qq.com  

Qi Liu  
School of Computer Science  
Hubei University of Technology  
Hubei, China  
260129443@qq.com

Abstract—Recently, most relational extraction models usually mitigate the adverse effects of noise in sentences for the prediction results, utilizing different tools of natural language processing that to capture high-level features in sentences combined. However, these attention mechanisms do not manage to exploit as much as possible the semantic information of certain keywords that have relational expressive information in the sentence. Therefore, this paper proposes a model based on the keyword’s attention mechanism, which is a novel attention mechanism based on the keywords of relational expression related. In particular, the proposed attention mechanism utilizes a linear-chain conditional random field that combines entity-pair features, similarity features between entity-pair features, and its hidden vectors to compute each word’s marginal distribution defined as the attention weight. Experimental results show that the method can focus on keywords with relational expression semantics in sentences without using sophisticated tools and achieves performance improvements on the SemEval-2010 Task 8 dataset.

Keywords—relation extraction; keywords attention; Hidden similarity; Bi-GRU

I. INTRODUCTION

RE(Relation extraction) is critical to NLP (natural language processing). To improve the performance of the models, researchers have tried several methods to remove the effects of noise [1, 2], including the removal of irrelevant words and methods based on attention mechanism. In contrast to traditional classification tasks, this task has to deal with noise in sentences.

In this paper, we propose an end-to-end bidirectional recurrent neural network-based model [3], called the REKA (RE Based on Keyword Attention) model, which uses a keyword-based attention mechanism. At the same time, our model avoids the accumulation of errors by not using any NLP tools, we use Gate Recurrent Unit (GRU) [4] to build a recurrent neural network to get the contextual information of the sentences. The keywords attention in the REKA model consists of two components: entity pair attention and segment attention, respectively. The paper utilized a linear-chain CRF (Conditional Random Field) [5] incorporating entity pair similarity calculations to calculate the marginal distribution of each state variable and consider it as an attention weight.

II. RELATED WORK

Recently, the models based on CNN (convolutional neural networks) [6, 7, 8] and RNN (recurrent neural network) [9] have become a major method for the RE research. The purpose of a CNN is to capture the local and continuous contextual content of a target, whereas an RNN accumulates contextual information in the input sentences via storage units. Socher et al. [10] proposed a RNN method that allows each node in the analysis tree to have a vector and a matrix, where the vector captures the intrinsic meaning of the component and the matrix captures how it changes the meaning of adjacent words or phrases. This matrix-vector RNN[10] can learn the meaning of operators in propositional logic and natural language, solving the problem that single word vector space models cannot capture the compositional meaning of long phrases, which prevents them from understanding language in greater depth.

Hashimoto et al. [11] proposed a RNN model based on syntactic trees in 2013. Unlike the model proposed by Socher et al. [10], Hashimoto et al. did not use word dependency matrices, which are computationally expensive, but used additional features such as lexical (Part-of-Speech) labels, phrase categories, and syntactic heads, and introduced into the RNN model Hashimoto’s model demonstrates the effectiveness of adding features and introducing averaging parameters to the RNN model to add weight to important phrases for the target task.

III. METHODOLOGY

As shown in Fig. 1, the proposed REKA is composed of the following four layers:

A. Input Layer

The input layer of the REKA model aims to convert the semantic and positional information of the input sentences into vectors. We use \( \{w_1, w_2, \ldots, w_n\} \) to denote the input sentences and \( \{p^{e_1}_1, p^{e_2}_2, \ldots, p^{e_j}_j\} \) denotes a vector of the relative position of every words to the entity pair \( e_j \) where \( j \in \{1, 2\} \).

To enable the model to capture more accurate semantic information, the paper used \( d_v \) dimensional ELMo (Embedding from Language Model) word embedding pre-training model. Unlike previous work in which one word corresponds to a

DOI reference number: 10.18293/SEKE2021-073
vector that is stationary [12, 13], ELMo word vectors are no more just a vector correspondence, a real trained model [14].

\[ \bar{h}_t = \text{GRU}(m_t) \quad (4) \]
\[ \bar{h}_t = \text{GRU}(m_t) \quad (5) \]
\[ h_t = [\bar{h}_t; h_t] \quad (6) \]

The input \( m_t \) of Bi-GRU is the output of the multi-head attention layer. To make effective use of both past and future features at a given time, the paper concatenates the hidden state vectors of the forward GRU network \( \bar{h}_t \in \mathbb{R}^{d_h} \) at each time step with the hidden state vectors of the backward GRU network \( h_t \in \mathbb{R}^{d_h} \), the \( d_h \) is the dimension of the hidden state vector of the GRU network unit, and we use \{\( h_1, h_2, \ldots, h_n \)\} to denote the hidden state vector of every word and use arrows to indicate the direction.

D. Keyword Attention

The keyword attention mechanism proposed in this paper aims to perform a soft selection of hidden layer vectors, and the attention weights are also a linear combination of a set of scalars. The weights are utilized to indicate the degree of attentions that the model should focus on a word of the sentence and it takes values between 0 and 1 in keyword attention mechanism. The proposed model defines a state variable \( z \), in which it means that the corresponding word is irrelevant to the relational classification when \( z \) is equal to 0, or the word required for the relational expression in the sentence if \( z \) is equal to 1. Thus each sentence has its corresponding sequence of binary state variables \( z \). According to this definition, the expected value of a hidden state \( N \), will be selected and is calculated as follows:

\[ N = \sum_i p(z_i = 1|H) h_i \quad (7) \]

To derive \( p(z_i = 1|H) \), the CRF is introduced here to calculate the sequence of weights for the hidden sequence \( H = \{h_1, h_2, \ldots, h_n\} \), in which \( H \) represents the input sequence and \( h_i \) represents the hidden output of GRU for the \( i \)th word. In particular, CRF provides a probabilistic framework for the computation of conditional probabilities from a sequence to another sequence. The linear-chain CRF defines a family of conditional probability \( p(z_i = 1|H) \) given \( H \) with the following equation:

\[ p(z_i \mid H) = \frac{1}{Z(H)} \prod_c \psi_i(z_i, H) \quad (8) \]
\[ Z(H) = \sum_{z \in \mathbb{Z}} \prod_c \psi_i(z_i, H) \quad (9) \]

Where \( \mathbb{Z} \) denotes the set of state sequences \( z \). \( Z(H) \) is the normalization constant and \( z_c \) denotes the subset of \( z \) given by individual clique \( c \). \( \psi_i(z_i, H) \) is the potential function of this clique shown as the following equation:

\[ \prod_{c \in C} \psi_i(z_c, H) = \prod_{i=1}^n \psi_1(z_i, h_i) \prod_{i=1}^{n-1} \psi_2(z_i, z_{i+1}) \quad (10) \]

This paper utilized two types of feature function for calculation, the vertex feature function \( \psi_1(z_i, H) \) and the edge feature function \( \psi_2(z_i, z_{i+1}) \). \( \psi_1 \) represents the mapping of the output \( h \) of GRU to the state variable \( z \), and \( \psi_2 \) simulates the transition
of two state variables at adjacent time steps. The equations for their definitions are shown as follows respectively:

\[
\psi_t(z, H) = \exp(\mathbf{W}^H F_t + \mathbf{W}^E F_2 + b) \tag{11}
\]

\[
F_1 = [h; p^0; h_{c_t}; t_2]; F_2 = [h_t; t_2; h_{c_2}; t_2] \tag{12}
\]

\[
\psi_z(z, z_{z+1}) = \exp(\mathbf{W}^z, z_{z+1}) \tag{13}
\]

Where \(\mathbf{W}^H\) and \(\mathbf{W}^E\) are learnable parameters of a linear transformation, and \(b\) is a bias term. They map the contextual information in the sentence into feature scores for each state variable, which makes use of the relative entity position features \(p^i\), \(p^2\) in the sentence and keyword features (entity pair features \(h_{c_1}\), \(h_{c_2}\) and entity pair hidden similarity features \(t_1\) and \(t_2\)).

1) Entity position feature: Relative location features \(p^i\), \(p^2\) are utilized to jointly represent contextual information as well as entity location relationships by concatenating them with the hidden layer outputs \(h_t\), as shown by \(F_1\) in Equation 12. There is a definition such as \(p^i, p^2 \in \mathbb{R}^{d_e}, e_j \in \{1, 2\}\).

Positional embedding is similar to word embedding vectors in which it transforms a relative positional scalar into a vector by traversing through the embedding matrix \(W_{pos} \in \mathbb{R}^{d_e \times (2L-1)}\), where \(L\) is the length of each piece of data in the dataset, and \(d_e\) is the dimension of the position vectors.

2) Entity hidden similarity features: Since entity words in sentences are inherently strong cues for relational classification, most of NLP tools were utilized in Zeng et al. [7] to obtain linguistic features of entity words, however, this approach is not an end-to-end model anymore. Therefore, this paper proposes a method to extract feature vectors that avoid the use of traditional NLP tools, and such features are named entity hidden similarity features in this paper, the calculation procedure is shown in Equation 14, 15.

\[
a_i = \frac{\exp \left( (h_j)^T c_i \right)}{\sum_{k=1}^K \exp \left( (h_j)^T c_k \right)} \tag{14}
\]

\[
t_{j \in \{1,2\}} = \sum_{i=1}^K a_i^j c_i \tag{15}
\]

In this paper, entity words are categorized based on the similarity between the embedding vector and the hidden vector of the entity words. Where \(c \in \mathbb{R}^{2d_e \times K}\) is a potential vector constructed in the potential vector space to represent the classes of similar entities, where \(K\) is the number of classes in which entities are classified by their hidden similarities.

The hidden similarity feature \(t\) of the \(j\)th entity is calculated by weighting the similarity \(c\) with the output \(h_{c_j}\) of the hidden layer based on the \(j\)th entity. Entity features are constructed by cascading the hidden states corresponding to the entity location and the potential type representation of the entity pair, as shown by \(F_2\) in Equation 12.

E. Classification Layer

To calculate the probability \(p\) of the output distribution of the state variable (the conditional probability of all relations), a softmax layer has been added after the keyword attention layer, which is calculated as shown in the Equation 16.

\[
p(y|\mathbf{N}) = \text{softmax}(\mathbf{W}_y \mathbf{N} + b_y) \tag{16}
\]

Where \(b_y \in \mathbb{R}^{|R|}\) is a bias term, \(|R|\) is a number of relationship categories, \(\mathbf{W}_y\) maps the expected value of a hidden state \(\mathbf{N}\) to the feature score for relation labels.

F. Training

With the introduction of the keyword attention mechanism, the model in this paper is shown in Fig. 1. This attention is calculated about to with concerning the cross-entropy loss of the RE. This loss function is defined as shown in Equation 17.

\[
\mathcal{L}' = -\sum_{i=1}^{2} \log p(y^{(i)}|S^{(i)}, \theta) \tag{17}
\]

Where \(|D|\) is the size of the training data set, \((S^{(i)}, y^{(i)})\) is the \(i\)th sample in the data set. In this paper, the AdaDelta optimizer is used to minimize the loss calculation parameter \(\theta\).

L2 regularization is added to the loss function to prevent overfitting, and \(\lambda_1, \lambda_2\) are hyper-parameters of regularizes. The second regularizer tries to force the model to process the few words that matter and returns a sparse weight distribution. The final objective function \(\mathcal{L}\) is shown in Equation 18.

\[
\mathcal{L} = \mathcal{L}' + \lambda_1 \| \theta \|^2 + \lambda_2 \sum_{i} p(z_i = 1|\mathbf{H}) \tag{18}
\]

IV. EXPERIMENTS

A. Dataset and Metric

Our experiments were evaluated on the SemEval-2010 Task 8 dataset. The dataset has 19 relationship types 10717 sentences, including 8000 samples for training and 2717 samples for testing. The evaluation metrics used in this paper are based on the macro-averaged F1.

B. Implementation Details

In this paper, the word embeddings used as input in the REKA model are trained using the publicly available pre-trained EMLo model, and all other parameters in the model are randomly initialized by the zero-mean Gaussian distribution, the hyperparameters are shown in Tab. 1.

C. Comparison Models

The proposed REKA model is compared with the following benchmark model such as SVM [6], MV-RNN [10], CNN [7], BLSTM [18], DepNN [19], FCM [20], SDP-LSTM [21].
TABLE I. HYPERPARAMETERS SETTING.

<table>
<thead>
<tr>
<th>Hyper-parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dropout rate</td>
<td>Keyword attention layer</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>Bi-GRU layer</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>Word embedding layer</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>Multi-head attention layer</td>
<td>0.8</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>Regularization coefficient</td>
<td>[0, 0.2]</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>Number of Heads</td>
<td>4</td>
</tr>
<tr>
<td>batch size</td>
<td>Size of mini-batch</td>
<td>50</td>
</tr>
<tr>
<td>$\eta$</td>
<td>Initial learning rate</td>
<td>4</td>
</tr>
<tr>
<td>$d_r$</td>
<td>The decay rate of leaning</td>
<td>0.5</td>
</tr>
<tr>
<td>$d_a$</td>
<td>Size of attention layer</td>
<td>50</td>
</tr>
<tr>
<td>$d_h$</td>
<td>Size of hidden layer</td>
<td>512</td>
</tr>
<tr>
<td>$K$</td>
<td>Number of the similar entities classes</td>
<td>4</td>
</tr>
<tr>
<td>$d_p$</td>
<td>Size of position embeddings</td>
<td>50</td>
</tr>
</tbody>
</table>

a. (The regularization coefficient values of $\lambda_1$ and $\lambda_2$ are selected from 0 to 0.2 using grid search.)

D. Experimental Results

To further evaluate the proposed model, we selected the RNN-based model from the above models for comparison. The average precisions (AP) of REKA compared with RNN methods are shown in Tab. II. The results of REKA model compared to other models are shown in Tab. III. From the experimental results, the proposed REKA model outperforms the existing model using the smaller number of features, with a relative improvement of 1.1%, indicating that the keyword attention mechanism can improve the performance of the model.

TABLE II. AVERAGE PRECISION SCORE FOR OUR MODEL AND COMPARABLE METHODS (MICRO-AVERAGED OVER ALL CLASSES)

<table>
<thead>
<tr>
<th>Model</th>
<th>BLSTM</th>
<th>SDP-LSTM</th>
<th>REKA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1%</td>
<td>0.26</td>
<td>0.47</td>
<td>0.55</td>
</tr>
<tr>
<td>20%</td>
<td>0.60</td>
<td>0.68</td>
<td>0.76</td>
</tr>
<tr>
<td>100%</td>
<td>0.73</td>
<td>0.70</td>
<td>0.81</td>
</tr>
</tbody>
</table>

a. (The first columns show how much of testing data has been used. Performance is on the SemEval-2010 task dataset)

TABLE III. COMPARATIVE RESULTS ON SEMEVAL-2010 TASK 8 TEST DATASET.

<table>
<thead>
<tr>
<th>Model</th>
<th>Additional Features</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM[6]</td>
<td>POS, WN, etc.</td>
<td>82.3</td>
</tr>
<tr>
<td>MV-RNN[10]</td>
<td>POS, NER, WN</td>
<td>82.4</td>
</tr>
<tr>
<td>CNN[7]</td>
<td>PE, WN</td>
<td>82.7</td>
</tr>
<tr>
<td>BLSTM[18]</td>
<td>None,</td>
<td>82.7</td>
</tr>
<tr>
<td>DepNN[19]</td>
<td>+ PF, POS, etc.</td>
<td>84.3</td>
</tr>
<tr>
<td>FCM[20]</td>
<td>DEP</td>
<td>83.6</td>
</tr>
<tr>
<td>SDP-LSTM[21]</td>
<td>SDP, NER</td>
<td>83.0</td>
</tr>
<tr>
<td>REKA Model</td>
<td>SDP</td>
<td>83.7</td>
</tr>
<tr>
<td></td>
<td>REKA</td>
<td>84.8</td>
</tr>
</tbody>
</table>

a. (Where WN, DEP, SDP, PE are WordNet, dependency features, shortest dependency path, position embeddings, respectively)

V. CONCLUSION

In this paper, we propose an end-to-end Bi-GRU network model based on a keyword attention mechanism. The model fully extracts the features available in the dataset using the keyword attention mechanism and achieves an F1 of 84.8 without the use of other tools for natural language processing. In the keyword attention mechanism, we use the relative position vectors of entity pairs and the similarity between entity pairs and their hidden vectors for computing the marginal distribution of each word, which is chosen as the attention weight. In the future, we will further investigate attention mechanisms that can better extract key information from sentences and plan to use them for the recognition of relationships between multiple entities.

REFERENCES

Chinese Sentence Semantic Matching With Multi-Granularity Based on Siamese Neural Network

Xuan Wen
School of Computer Science
Hubei University of Technology
Hubei, China
1595159972@qq.com

Jianxia Chen
School of Computer Science
Hubei University of Technology
Hubei, China
chenjianxiawh@gmail.com

Yu Chen
School of Computer Science
Hubei University of Technology
Hubei, China
1148848330@qq.com

Shirui Sheng
School of Computer Science
Hubei University of Technology
Hubei, China
1606472332@qq.com

Abstract—Sentence semantic matching is one of critical research in various NLP tasks such as natural language inference, paraphrase identification, and question answering, in which similarity of input sentences has always been a key aspect to determine the semantic relations of sentences. One of the most popular models is to utilize single word granularity to address the semantic similarity. However, it is not appropriate for Chinese sentences semantic matching. This is because there are various meanings following various granularities such as characters or word segmentation in a Chinese sentence. In addition, it is difficult for the sentence semantic matching due to its own short contents and sparse features. Inspired by Siamese Neural Network, an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors, this paper proposes a Multi-Granularity Fusion neural network, which enables preserving semantic features from both the character-granularity and the word-granularity in Chinese sentences. The paper evaluates the proposed architecture on highly competitive benchmark datasets related to sentence matching. Experimental results show that the proposed architecture, which retains both characters and words features of sentences, and achieves state-of-the-art performances for most of the tasks.

Index Terms: Siamese Network; Multi-Granularity; Chinese Sentence; Semantic Matching

I. INTRODUCTION

Semantic matching plays a critical role in many NLP (natural language processing) tasks, such as question and answer (QA) [1], machine translation (MT) [2], information retrieval [3], etc. As a widely existing text representation over the Internet, sentence semantic matching has been gradually showing its strong research values. How to effectively excavate and analyze the sentence semantic has become a research hotspot in the field of NLP.

Usually, the most serious issues of sentences semantic matching are resulted from their short contents and sparse features. To solve this problem, traditional methods often mine information according to the original text, such as using semantic dictionary HowNet [4] or introducing topic model LDA [5] to assist sentences semantic matching. However, these methods often carry out semantic matching only by weighting the method of the original sentences. Obviously, the rich semantic information of the sentences in Chinese texts has not been fully utilized to improve the matching performance. This is because Chinese texts have its own particularity, the semantic matching of Chinese sentences is often affected by their word segmentations.

The common semantic matching methods usually calculated similarities from the aspect of the word granularity, however, ignored the extraction of semantic features of the sentence. With the rapid development of deep learning, many deep learning models based on the word granularity have been proposed in the research of sentence semantic matching, such as DeepMatch tree [6], Match-Pyramid [7], ARC-I [8], etc. However, these models do not obtain the rich characteristics of the sentence itself via only the single word granularity. It may even become worse only depending on a single word granularity calculation. Therefore, some researchers design a novel neural network model that can combine words with characters together to form a new sequence such as Lattice CNNs [9]. Although this network has achieved better results in QA (question and answer), such a simple combination may introduce the noise and even lose its original meaning.

Inspired by SNN (siamese neural network) [10], an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors, this paper proposes a novel architecture, named MGFSN which is an abbreviation for Multi-Granularity Fusion Siamese neural Network. The MGFSN enables preserving semantic features from both the character-granularity and the word-granularity in Chinese sentences. In particular, the proposed MGFSN architecture is composed of three components including the word embedding layer, the multi-granularity coding layer, and the semantic interaction layer. The paper evaluates the proposed architecture on a highly competitive benchmark LCQMC dataset related to sentence matching. Experimental results show that the proposed architecture achieves state-of-the-art performances for most of the tasks.

The rest of this paper is structured as follows. The paper introduces the related work about semantic matching of Chinese sentences in Section II, and describes the proposed...
Multi-Granularity Fusion based on the Siamese Network model in Section III in detail. Section IV provides the experimental results and related analysis, Section V summarizes the contributions of the paper and the future work.

II. RELATED WORK

A. Sentence similarity calculation

Sentence similarity calculation is the basis of natural language understanding tasks. Sentence similarity refers to the degree of interchangeability of words between two sentences and the degree of consistency of word meaning [11], which is an index used to evaluate the sentences similarity. From the perspective of information theory, Lin et al. [12] believe that sentence similarity is related to the commonality among sentences. The greater the commonality, the smaller the difference and the higher the similarity. Therefore, the calculation of similarity of two sentences, including $S_1$ and $S_2$, shown in Eq. (1).

$$\text{sim}(S_1, S_2) = \frac{\log P(\text{common}(S_1, S_2))}{\log P(\text{description}(S_1, S_2))}$$

Since there are many infect factors in the sentence similarity calculation such as sentence structure, language, syntax, etc., there are various ways to research sentence similarity. The classification methods recognized by most scholars are string-based method, corpus-based method, and knowledge-based Library methods, knowledge-based methods and hybrid methods [13-15]. Among them, this paper utilizes a corpus-based method, specifically a neural network-based method. The neural network-based method takes a corpus to convert a sentence into a vector representation with semantic information as in input for learning. Compared with other methods, the biggest advantage of this method is that it can represent complex contexts.

B. Convolutional Neural Network

Convolutional neural network (CNN) is a classic deep learning algorithm. Its basic idea is to use parallel multi-level convolution to perform multi-layer representation of input data, extract feature information of data, and obtain better feature robustness. It was first used in computer vision (CV). With its mature application in the field of computer vision, people have also begun to apply it to text processing, such as the model proposed by Kim et al. in 2014 [16].

After the first success of Convolutional Neural Network in NLP field, more and more people apply CNN in NLP field. Kal et al. [17] proposed a Network model named DCNN (Dynamic Convolutional Neural Network), whose delicacy lies in the use of Dynamic pooling method which can process input of variable length. The network contains two types of layers, namely the one-dimensional convolutional layer and the dynamic k-max pooling layer. The structure of DCNN is shown in Fig. 1. The convolution layer of the network adopts the way of wide convolution, followed by the dynamic k-max pooling layer, which retains the first k maximum values with certain position information. Then, the pooled features are folded, mainly to consider some relation between two adjacent rows, and the model adopts the RAE model idea to extract features hierarchically. The advantage of DCNN is that it does not need any prior information input, nor does it need to construct very complex artificial features.

![Figure 1. The Framework Of DCNN [17]](image)

Wang et al. [18] proposed a network structure based on similar and dissimilar information, which considered the similarity and dissimilarity of sentences by decomposing and combining the semantics of words, and decomposed two sentences into similar matrix and dissimilarity matrix. He et al. [19] proposed a network structure with multiple perspectives and granularity, which fully excavates the characteristic information of the sentence and improves the performance of the model, but at the same time it makes the model more complex and time-consuming. Ma et al. [20] took into account the dependency information of the sentence and integrated the dependency information into the sentence.

Although the above methods have made some progress in the application of CNN, there is still a lack of consideration of sentence granularity and the problem of long time consumption. Therefore this paper proposes the corresponding sentence vector, extract the character granularity and the word granularity feature respectively.

C. Siamese Neural Network

Siamese neural network is a neural network architecture composed of two or more identical subnets, which is widely used in the task of determining the consistency of two kinds of data and measuring the relationship between things [21-23]. One of the architecture of Siamese network is shown in Fig. 2.
The parameters and weights are shared among the subnets, and the parameters are updated at the same time. The main idea is to use the network or function to map the input to the target space, and then use distance calculation formulas such as cosine distance or Euclidean distance to compare the similarity in the target space. If the mapping network or function is $G_w(X)$ and the parameter is $W$, the similarity measurement result is:

$$E_w(x_i, x_j) = f(G_w(x_i), G_w(x_j))$$ (2)

Due to the sharing parameters between Siamese network subnets, the proposed model training requires fewer parameters, which means that less data is required to train the model to reduce the possibility of over-fitting.

III. MGFSN MODEL

A. Framework Overview

As shown in Fig. 3, the proposed MGFSN architecture is composed of the three components: (1) the word embedding layer, (2) the multi-granularity coding layer, and (3) the semantic interaction layer.

The paper denotes two input Chinese sentences as $P = \{ P_{w1}, P_{w2}, ..., P_{wi}, ..., P_{w5} \}$ and $Q = \{ Q_{w1}, Q_{w2}, ..., Q_{wi}, Q_{w6}, ..., Q_{w7} \}$, where $w_i$ is the $i^{th}$ word of the sentence $P/Q$, $i$ is the word length of $P/Q$, $c_j$ is the $j^{th}$ character of the sentence $P/Q$ and $j$ is the character length of $P/Q$.

B. Embedding Layer

To construct the appropriate sequence representation, the paper concatenates words embedding including both Chinese words segment representations and characters representations. Using jieba tool [25] to segment sentences, the paper obtains the sequence of the word granularity, and divides it directly through characters to obtain the sequence of the character granularity.

In the word embedding, each word is represented as a d-dimensional vector by using a pre-trained word embedding method such as Word2Vec [26]. In the MGFSN model, a word embedding vector can be divided into two types including word granularity and character granularity.

C. Multi-Granularity Fusion Encoding Layer

As the most critical component in MGFSN model, multi-granularity fusion encoding layer, named MGFE layer in abbreviation, will extract features both of word granularity and char granularity via the SNN, in which no external resources are being introduced thus the semantic coding performance can be improved effectively.

As illustrated in Fig. 4, the proposed MGFE layer consists of two different Siamese network that have the same network structure with various weight training. Each Siamese network consists of two identical sub-networks in which $P-e_i/Q-e_i$ and $P-e_i/Q-e$ means the results of sentence $P/Q$ passing through the embedding layer.

Firstly, for word vector, this paper utilizes self-attention as an attention mechanism and chooses dot product to calculate the attention matrix in the equation (3-4):

$$f(m_i, m_j) = m_i^T m_j$$ (3)

$$a_i = Attention(m_i, m_j) = softmax(f(m_i, m_j)m_i$$ (4)

Where $m_i = m_{w} = Word_{vector} \cdot f(.)$ is the matmul operation and softmax is normalized difference index function. This is done after the vectors in order to fully consider the semantic and grammatical connections between the different words in the sentences. And then extracts its features through two convolution neural networks in the equation (5-8):

$$c_i = Conv(a_i)$$ (5)

$$m_i = MaxPool(c_i)$$ (6)
\[ c_i = \text{Conv}(m_i) \]  
\[ m_j = \text{MaxPool}(c_j) \]

Where \( a_i \) is the vector after Attention function, \( c_i \) is the the result of \( i \)th convolution function and \( m_j \) is the result of \( j \)th MaxPool function.

Meanwhile, for the character vectors, this paper adopts the same network structure with the same operation. Finally, both the character and word granularity are concatenated to obtain more semantic representation information.

\[ |S_P Q| = \text{concatenate}(|P - Q|, |P \times Q|) \]  
\[ |S_P Q| = \text{Concatenate}(|S, M|) \]

Where \( |P, Q| \) is the output of multi-granularity fusion encoding layer, \( |S, M| \) is the absolute value of \( |P - Q| \) and \( |P \times Q| \), \( |S, M| \) is the result of concatenating \(|S, M|\).

As shown in Fig. 5, \( P\text{-feature} \) and \( Q\text{-feature} \) are handled by Eq. (9) and Eq. (10) to obtain the vector \( S \) and \( M \). Then the paper concatenates both vector \( S \) and vector \( M \) via Eq. (11). After that, the results of concatenating are extracted using two dense layers, whose dimensions are 256 and 512 respectively. At the same time, \( P\text{-feature} \) and \( Q\text{-feature} \) are extracted by two dense layers, whose dimensions are 256 respectively. Afterward, this paper adds the two vectors resulted from the above operation with the superposition effect to generates the final matching representation of input sentences, the matching degree, which will be transferred into Sigmoid function.

IV. EXPERIMENT AND ANALYSIS

A. Data-set

The data set used in this paper is LCQMC [27], which has a large-scale Chinese question matching corpus contains 260,068 problem pairs with manual annotations. In this paper, it is divided into three parts with the same proportion as in [27], that is, the training set containing 238,766 problem pairs, the development set containing 8,802 problem pairs, and the test set containing 12,500 problem pairs.

Illustrated in TABLE I, each data sample has three attributes: “sentence1”, “sentence2” and “Label”, sentence 1 and sentence 2 are text pairs. If Label is equal to 1, it means that the semantics for sentence 1 and sentence 2 is similar, or 0, it means the semantics for sentence 1 and sentence 2 are not similar.

<table>
<thead>
<tr>
<th>Sentence Pairs</th>
<th>Semantic Match</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1:求一款网页游戏</td>
<td>1</td>
</tr>
<tr>
<td>Q2:找一款网页游戏</td>
<td>0</td>
</tr>
<tr>
<td>Q3:在家带小孩怎么赚钱</td>
<td></td>
</tr>
<tr>
<td>Q4:有什么工作适合在家带孩子做的</td>
<td></td>
</tr>
<tr>
<td>Q5:What kind of jobs are suitable for stay at home parenting</td>
<td></td>
</tr>
</tbody>
</table>
B. Experimental environments

The proposed MGFSN model implements all experiments on a 2080Ti GPU with 11G explicit memory programming by Python based on the Keras and TensorFlow2.0 framework. The parameters are defined as follows.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Embedding layer</td>
<td>300</td>
</tr>
<tr>
<td>CNN filters</td>
<td>128</td>
</tr>
<tr>
<td>kernel size</td>
<td>3</td>
</tr>
<tr>
<td>activation</td>
<td>Tanh</td>
</tr>
<tr>
<td>Dropout</td>
<td>0.3</td>
</tr>
<tr>
<td>Maxpooling</td>
<td>3</td>
</tr>
<tr>
<td>Batch size</td>
<td>512</td>
</tr>
<tr>
<td>Loss function</td>
<td>binary_crossentropy</td>
</tr>
</tbody>
</table>

C. Baseline & Metric

Liu et al.[27] have implemented eight relevant and representative state-of-the-art methods in LCQMC. Those methods have been used as baselines for evaluating the models in this paper.

Unsupervised Methods: word mover distance (WMD), word overlap (Cwo), n-gram overlap (Cngram), edit distance (Dedt), and cosine similarity respectively (Scos) [27].

Supervised Methods: convolutional neural network (CNN), bidirectional long short term memory (BiLSTM), bilateral multi-Perspective matching (BiMPM) [27].

This paper evaluates the Accuracy, Precision, Recall, F1 of all methods. Before calculating, this paper defines: True Positive is abbreviated as TP, FP is abbreviated as False Positive, TN means True Negative, FN means False Negative.

So the calculation formulas are described in the equations (12-15) as follows:

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (12)
\]
\[
\text{Precision} = \frac{TP}{TP + FP} \quad (13)
\]
\[
\text{Recall} = \frac{TP}{TP + FN} \quad (14)
\]
\[
\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (15)
\]

High accuracy and F1-score indicate better performance of the model. Both of them are used in this paper.

D. Performance comparison

Compared to unsupervised methods as shown in TABLE III, WMDchar, WMDword, Cwo, Cngram, Dedt, Scos, the proposed model MGFSN improves the precision metric by 34.75% at the highest and 14.25% at the lowest, recall by 10.87% at the highest and 0.17% at the lowest, F1-score by 24.67% at the highest and 11.77% at the lowest and accuracy by 32.11% at the highest and 13.71% at the lowest.

In contrast to the unsupervised approach, the proposed MGFSN model is a supervised model. MGFSN model can use the error between the real label and the prediction to carry out backpropagation, so as to correct and optimize the massive parameters in the neural network. In addition, since MGFSN model uses multiple granularities, there are more features that are good for similarity judgment. Therefore, the MGFSN has made great progress compared with the unsupervised method.

Compared with the supervised and neural network approach as shown in TABLE III, CBOWchar, CBOWword, CNNchar, CNNword, BiLSTMchar, BiLSTMword, BiMPMchar, BiMPMword, MGFSN improves the precision metric by 14.75% at the highest and 3.55% at the lowest, recall by 6.67% at the highest and -4.43% at the lowest, F1-score by 11.37% at the highest and 0.17% at the lowest and accuracy by 13.81% at the highest and 1.01% at the lowest.

In contrast to the above supervised and neural network approach, MGFSN model not only uses multiple granularity to obtain richer features, but also can extract richer and deeper semantic features because of its deeper network structure.

Figure 6. The Histogram of Experiments on LCQMC Sorted by Accuracy
TABLE III. EXPERIMENTS ON LCQMC

<table>
<thead>
<tr>
<th>Methods</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMDchar</td>
<td>67.0</td>
<td>81.2</td>
<td>73.4</td>
<td>70.6</td>
</tr>
<tr>
<td>WMDword</td>
<td>64.4</td>
<td>78.6</td>
<td>70.8</td>
<td>60.0</td>
</tr>
<tr>
<td>Cwo</td>
<td>61.1</td>
<td>83.6</td>
<td>70.6</td>
<td>70.7</td>
</tr>
<tr>
<td>Cgram</td>
<td>52.3</td>
<td>89.3</td>
<td>66.0</td>
<td>61.2</td>
</tr>
<tr>
<td>Ddct</td>
<td>46.5</td>
<td>86.4</td>
<td>60.5</td>
<td>52.3</td>
</tr>
<tr>
<td>Scos</td>
<td>60.1</td>
<td>88.7</td>
<td>71.6</td>
<td>70.3</td>
</tr>
<tr>
<td>CBOWchar</td>
<td>66.5</td>
<td>82.8</td>
<td>73.8</td>
<td>70.6</td>
</tr>
<tr>
<td>CBOWword</td>
<td>67.9</td>
<td>89.9</td>
<td>77.4</td>
<td>73.7</td>
</tr>
<tr>
<td>CNNchar</td>
<td>67.1</td>
<td>85.6</td>
<td>75.2</td>
<td>71.8</td>
</tr>
<tr>
<td>CNNword</td>
<td>68.4</td>
<td>84.6</td>
<td>75.7</td>
<td>72.8</td>
</tr>
<tr>
<td>BILSTMchar</td>
<td>67.4</td>
<td>91.0</td>
<td>77.5</td>
<td>73.5</td>
</tr>
<tr>
<td>BILSTMword</td>
<td>70.6</td>
<td>89.3</td>
<td>78.9</td>
<td>76.1</td>
</tr>
<tr>
<td>BIMPMchar</td>
<td>77.6</td>
<td>93.9</td>
<td>85.0</td>
<td>83.4</td>
</tr>
<tr>
<td>BIMPMword</td>
<td>77.7</td>
<td>93.5</td>
<td>84.9</td>
<td>83.3</td>
</tr>
<tr>
<td>MGFSN</td>
<td>81.25</td>
<td>89.47</td>
<td>85.17</td>
<td>84.41</td>
</tr>
</tbody>
</table>

V. CONCLUSION

Here is an explanation to a novel approach based on Siamese Network with Multi-Granularity Fusion for Chinese sentence semantic matching. The MGFSN model in this paper is based on Siamese architecture which reduces the parameters of the model, calculating not only the word granularity of Chinese but also the character granularity of Chinese. In particular, multi-granularity fusion is utilized to obtain more features for similarity matching.

Extensive experiments are carried out on the latest similarity matching benchmark LCQMC. Experimental results show that the proposed approach achieves excellent performances for most of the tasks. In the future, the paper would like to look for various features granularities such as clauses to enrich the features of the sentence, or take different other pre-trained contextual embeddings such as ELMo or BERT to improve performance of the approach.

REFERENCES

Complementary Representation of ALBERT for Text Summarization

Wenyeng Guo, Bin Wu, Bai Wang, Lianwei Li, Junwei Sun, Maham Nazir
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia
Beijing University of Posts and Telecommunications, Beijing, China
{gwy, wubin, wangbai, llw, junweisun, maham}@bupt.edu.cn

Abstract—Pretraining has proved to be an effective strategy to learn the parameters of the deep neural network. It captures the world knowledge that can be adapted to downstream tasks. Text summarization based on ALBERT [1] outperformed previous work by a large margin. However, they only use the final layer as a contextualized representation of the input text. Multiple studies have proven that intermediate layers also encode the rich hierarchy of linguistic information. In this paper, we propose a Fast Complementary Representation Network (FCRN), which dynamically incorporates linguistic knowledge spread across the entire ALBERT for extractive selection. Different from previous work, we measure the importance of hidden layers by all sentence representations rather than all token embeddings, which can filter nonsignificant words and takes six times less time during training. FCRN first obtains the importance of each layer by sentence embeddings and then automatically absorbs the supplementary information to ALBERT’s output. We conduct experiments on CNN/DailyMail and XSum datasets. The results show that our model obtains higher ROUGE scores.

Index Terms—Text summarization, Pretrained Language Model, Fast Complementary Representation network

I. INTRODUCTION

Text summarization is the task of condensing the essence of a document to converse the main information contents and main ideas quickly. There are two paradigms to generate a summary: extractive summarization and abstractive summarization. Extractive summarization generates the summary by scoring and extracting the high-score sentences (or more fine-grained units) from source documents generally, such as [2], [3]. In contrast, abstractive summarization is an ideal form of summarization since it generates the novel sentence required to have recourse to world knowledge. Abstractive summarization generates a summary in a human-written way.

Extractive summarization remains more reliable than abstractive summarization since the summary is composed of the text span (like sentence) in the source document. In this paper, we primarily focus on extractive summarization. The pretrained language model has recently brought Natural Language Processing (NLP) to a new era since the emergence of BERT [4].

Numerous summarization approaches have achieved state-of-the-art results based on the pretrained language models. BERTSUMEXT [5] built a BERT-based minimum-requirements model, which outperformed previous work by a large margin. DISCOBERT [6] extracted sub-sentential discourse units based on BERT and Graph Convolutional Network. MATCHSUM [7] adopted a Siamese-BERT architecture to select the candidate summary, which was the most similar to the target summary. TSEM [8] showcased the effectiveness of ALBERT to handle text summarization. However, the improvement on automatic metrics like ROUGE has reached a bottleneck. They only use the final layer of the pretrained language model as contextualized representations of the input text. The popularity of transformer-based models has driven researchers to study what was behind their success.

Multiple studies [9]–[11] have proven that transformer-based models composed a hierarchy of linguistic signals ranging from surface to semantic features. Intermediate layers also encode the rich hierarchy of linguistic information. Taking the output of the pretrained language model restricts the power of pretrained representation when fine-tuning [9], [12], [13]. ALBERT has fewer parameters and focuses on modeling intersentence coherence. Meanwhile, TSEM has achieved better performance based on ALBERT than BERTSUMEXT based on BERT in the summarization task. In this paper, we propose an architecture stacked on ALBERT, namely Fast Complementary Representation Network (FCRN), which can leverage the rich linguistic knowledge to enhance the power of ALBERT. There is no research to study the effectiveness of the fused representation of the ALBERT on text summarization. In our model, the ALBERT’s embeddings of hidden layers are complementary to the output of ALBERT. To capture the information hierarchically, different from previous work [12], [13], we first obtain the sentence embedding in the document to capture the gist of the individual sentence. The number of sentences is far more than tokens, which can reduce training time. The embeddings of sentences that contain the main information are input to FCRN to catch a complementary representation of the final output. We highlight our contributions as follows:

- We proposed FCRN to capture more language information of sentence. We also study the performance of different language knowledge on text summarization. To our knowledge, we are the first ones to dynamically fuse the rich information spread across the entire model on ALBERT.
- In our model, We propose taking sentence embeddings rather than all token embeddings as the input of FCRN.
It shortens the training and inference time with better performance. Our model can generate a better ground-truth to train the MATCHSUM.

- We identify the importance of linear word order, syntactic rule, and semantic information for the summarization task.

II. RELATED WORK

A. Pretrained Language Model

The state-of-the-art result has been achieved by pretrained language models in many NLP tasks. Pretrained language models have advanced downstream NLP tasks by learning universal language representation. In earlier research, the pretrained language model aims to learn good word embeddings containing semantic meaning. Those pretrained representations are added to downstream tasks as additional features. For BERT, ALBERT, and RoBERTa [14], they obtain the embedding of the token with context. Unlike the earlier pretrained language models, they can capture syntactic structures, semantic rules, and context-dependent natures of words. The representation of a given token in vocabulary depends on the whole text. Those models are stacked with a deep network, fine-tuned on downstream tasks with a better model initialization.

Currently, the transformer-based pretrained language model has drawn more attention. The best known transformer-based model is BERT. The curiosity about transformer-based models has driven over 150 studies of the popular BERT model. Based on this knowledge, many enhanced versions of BERT are proposed. The backbone of their architecture is similar to BERT. RoBERTa improves BERT by dynamic masking. ALBERT replaces the next sentence prediction with sentence order prediction to model inter-sentence coherence. All of them can be fine-tuned to adapt to special downstream tasks.

B. Abstractive Summarization

Neural abstractive summarization models conceptualize the task as a sequence-to-sequence problem, where an input sequence is mapped into another output sequence. In 2015, Rush et al. [15] applied the neural encoder-decoder architecture to abstractive summarization and thus paved the way for using neural networks for abstractive summarization. Nallapati et al. [16] and See et al. [17] applied a pointer generator network which generates words from a fixed vocabulary or copies from the source document, which is an effective method to handle Out of Vocabulary (OOV). See et al. presented a coverage mechanism to discourage repetition. To solve the mismatch between the learning object and the evaluation criterion, reinforcement learning-based models trained by optimizing the ROUGE metric achieved higher performance. Encoder-decoder transformers have shown great successes for abstractive summarization.

C. Extractive Summarization

Extractive summarization has gained more attention with its simplicity and facticity. It is often defined as a binary classification. The label of the text span indicates whether it should be included in the summary. Initially, statistical methods generate summaries leveraging the similarity between sentences. They consider statistical features, including sentence position, term frequency (TF), and the inverse document frequency (IDF).

Neural networks are primarily introduced to text summarization by modeling the semantic meaning of sentences [18]. Recently, deep neural networks have achieved great success in summarization tasks. Neural text summarization generally obtains the sentence representation by a neural encoder. Nallapati et al. [19] instantiated the encoder by recurrent neural network (RNN). Zhong et al. [20] leveraged transformer to encode the semantic meaning by interacting between sentences.

Graph Neural Networks (GNN) can learn from complex structured data. Xu et al. [6] presented how GNN can be usefuly applied in text summarization. NeuSUM [21] was a neural extractive document framework that jointly learnt to score and select sentences. The reinforcement learning-based model trained by directly optimizing the ROUGE metric achieved state-of-the-art results [22], [23]. Pretrained language models have achieved state-of-the-art results on extractive summarization. HibERT [3], BertSUMEXT, MATCHSUM are based on BERT, and TSEM [8] explored the potential of ALBERT on text summarization. It showed ALBERT performed better than BERT with fewer parameters. The pretrained language model was fine-tuned to adapt to text summarization in the above work. They only use the final layer as a contextualized representation of the input text.

Compared to the models outlined above, we explore the potential of fusing the representation of multiple layers as complementary to the last layer. It makes sentence embeddings capture more language knowledge. Moreover, sentence embeddings are used to weight all hidden embeddings which can take less time and capture the sense of sentence.

III. MODEL

A. Overview

Fig. 1 presents an overview of our model, which consists of a Sentence Encoder and FCRN. For the Sentence Encoder, a pretrained ALBERT is used to output the representation of each sentence in all hidden layers. FCRN takes the output of the Sentence Encoder as input and dynamically summarizes the hidden representation based on the output of Bi-LSTM, which enhances the power of the final output. The outputs of FCRN are used to predict the label of the sentence.

Let D = [sent_1, sent_2, ..., sent_n] denote a document with n sentences, where sent_i is the i-th sentence. We formulate extractive summarization as a sequence labeling task, in which the label sequence Y = [y_1, y_2, ..., y_n] ∈ {0, 1} indicates whether the corresponding sentence is included in the summary or not. We will calculate the final predicted score ŷ_i. The loss of our model is the binary classifier entropy between the prediction and the gold label.

\[
L = - \sum_{i=1}^{n} (y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))
\]
Figure 1. (left) is an overview of our model, FCRN is stacked on ALBERT encoder. (right) is the architecture of FCRN. $\sigma$ is a softmax function to calculate the importance of each layer.

### B. Sentence Encoder

ALBERT is a transformer-based encoder, which can generate the contextual representation of words based on tokens, segment tokens, and position ids. ALBERT is pretrained on a sentence or sentence pair, and a special classification token ([CLS]) is used as the aggregate representation for the classification task. Given Document D is a document with sentences, Document D is a document with multiple sentences needing to be classified, we insert [CLS], [SEP], [CLS], and [SEP] at the beginning and the ending of each sentence. [CLS] is to capture the meaning of an individual sentence and [SEP] is viewed as the boundary of the sentence. Segment token {0, 1} indicates the position of sentences is odd or even in the document. For document D, the input tokens are [CLS], $s_{sent1}$, [SEP], $s_{sent2}$, [SEP], ..., [CLS], $s_{sentn}$, [SEP]], and the segment tokens are [0, 1, 0, 1, ...]. We input D as the input to ALBERT like in Fig. 1. All hidden-state embeddings are computed by ALBERT, The formula is as follows:

$$T_1, ..., T_l = \text{ALBERT}([\text{CLS}], s_{sent1}, [\text{SEP}], ..., [\text{CLS}], s_{sentn}, [\text{SEP}])$$  \hspace{1cm} (2)

$$s_1, ..., s_l = \text{Extractor}(T_1, ..., T_l)$$  \hspace{1cm} (3)

where $T_i$ is the output of the $i$-th layer, and $l$ denotes the number of layers in ALBERT. However, $T_i$ contains the embedding of each token in the document. We select the embeddings of [CLS] as the sentence embeddings by the Extractor in each layer. The sentence embeddings of $i$-th layer $s_i \in \mathbb{R}^{n \times d}$, where $n$ is the number of sentences and $d$ is the hidden size of the encoder. They will be sent to FCRN to generate the final sentence representation.

### C. FCRN

FCRN is proposed to generate a more powerful representation. We can select a different number of input layers for FCRN. To capture the long-range dependency of sentences, a single-layer bidirectional Long Short Time Memory (Bi-LSTM) is used to weight all hidden sentence embedding. We apply Layer Normalization (LayerNorm) to Bi-LSTM to stabilize the training. The formulation of Bi-LSTM are as follows:

$$\begin{pmatrix}
F_i \\
I_i \\
O_i \\
C_i
\end{pmatrix} = L_i \left( W_{hi} H_{i-1} + W_{xi}s_i \right)$$  \hspace{1cm} (4)

$$C_i = \sigma(F_i) \odot C_{i-1} + \sigma(I_i) \odot \tanh(G_{i-1})$$  \hspace{1cm} (5)

$$H_i = \sigma(O_i) \odot \tanh(L_{N_c}(C_i))$$  \hspace{1cm} (6)

Where $H_i$ is the hidden state at time $i$, $C_i$ is the cell state at the time $i$, and $I_i$, $F_i$, $C_i$, $O_i$ are the input, forget, cell, and output gates, $L_{N_h}$, $L_{N_x}$, $L_{N_c}$ are different layer normalization operations. The bias is ignored in the formulation. $x_i$ is the input in step $i$.

The input to the FCRN is $s_1, s_2, ..., s_l$. We take $s_i$ to Bi-LSTM. Bi-LSTM will obtain a fixed-sized embedding $K_i$ to capture the knowledge in $i$-th layer. We take the concatenation
TABLE I
THE RESULTS OF OUR PROPOSED MODELS ON CNN/DAILYMAIL DATASET. MODELS WITH SUBSCRIPT "w/o" WERE TRAINED AND TESTED ON THE ANONYMIZED-VERSION DATASET. FCRN \(_{(w/o)} \) \(^*\) MEANS THE MODEL WITHOUT (WITH) \(^*\) LAYERS.

<table>
<thead>
<tr>
<th>Model</th>
<th>R1</th>
<th>R2</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE</td>
<td>51.64</td>
<td>30.48</td>
<td>47.88</td>
</tr>
<tr>
<td>LEAD-3</td>
<td>40.37</td>
<td>17.44</td>
<td>36.61</td>
</tr>
<tr>
<td>Run-ext+RL(^*)</td>
<td>40.55</td>
<td>18.42</td>
<td>36.84</td>
</tr>
<tr>
<td>NeuSUM</td>
<td>41.86</td>
<td>19.16</td>
<td>38.20</td>
</tr>
<tr>
<td>BertSUM/ext</td>
<td>43.32</td>
<td>20.32</td>
<td>39.71</td>
</tr>
<tr>
<td>FCRN(^{com})</td>
<td>43.43</td>
<td>20.36</td>
<td>39.78</td>
</tr>
<tr>
<td>TSEM-sentence</td>
<td>43.54</td>
<td>20.47</td>
<td>39.93</td>
</tr>
<tr>
<td>FCRN</td>
<td>43.66</td>
<td>20.54</td>
<td>40.03</td>
</tr>
<tr>
<td>MATCH-ORACLE</td>
<td>51.08</td>
<td>26.94</td>
<td>46.99</td>
</tr>
<tr>
<td>MATCH-ORACLE-FCRN</td>
<td>51.29</td>
<td>27.15</td>
<td>47.39</td>
</tr>
<tr>
<td>MATCHSUM-BERT</td>
<td>44.22</td>
<td>20.62</td>
<td>40.38</td>
</tr>
<tr>
<td>MATCHSUM-FCRN</td>
<td>44.36</td>
<td>20.70</td>
<td>40.51</td>
</tr>
<tr>
<td>FCRN(^{w/o} high)</td>
<td>43.48</td>
<td>20.36</td>
<td>39.85</td>
</tr>
<tr>
<td>FCRN(^{w/o} middle)</td>
<td>43.34</td>
<td>20.27</td>
<td>39.71</td>
</tr>
<tr>
<td>FCRN(^{w/o} low)</td>
<td>43.61</td>
<td>20.51</td>
<td>40.01</td>
</tr>
<tr>
<td>FCRN(^{w} high)</td>
<td>43.28</td>
<td>20.22</td>
<td>39.65</td>
</tr>
<tr>
<td>FCRN(^{w} middle)</td>
<td>43.64</td>
<td>20.53</td>
<td>40.04</td>
</tr>
<tr>
<td>FCRN(^{w} low)</td>
<td>43.05</td>
<td>19.98</td>
<td>39.40</td>
</tr>
</tbody>
</table>

TABLE II
THE RESULTS OF OUR PROPOSED MODELS ON XSUM DATASET.

<table>
<thead>
<tr>
<th>Model</th>
<th>R1</th>
<th>R2</th>
<th>RL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORACLE</td>
<td>28.90</td>
<td>8.19</td>
<td>21.86</td>
</tr>
<tr>
<td>LEAD</td>
<td>19.66</td>
<td>2.39</td>
<td>14.80</td>
</tr>
<tr>
<td>BertSUMext</td>
<td>23.53</td>
<td>4.54</td>
<td>17.80</td>
</tr>
<tr>
<td>TSEM-sentence</td>
<td>23.91</td>
<td>4.69</td>
<td>18.11</td>
</tr>
<tr>
<td>FCRN</td>
<td>24.14</td>
<td>4.74</td>
<td>18.27</td>
</tr>
<tr>
<td>MATCH-ORACLE</td>
<td>29.54</td>
<td>7.40</td>
<td>22.55</td>
</tr>
<tr>
<td>MATCH-ORACLE-FCRN</td>
<td>30.10</td>
<td>7.38</td>
<td>23.08</td>
</tr>
</tbody>
</table>

of each direction’s final state as a knowledge of the current layer.

\[ K_i = ReLU (LayerNormLSTM (s_i)) \]  

The importance of each layer will be calculated as:

\[ \alpha_i = softmax (K_i) \]  

Given the importance of each layer, we calculate the complementary representation \( Com \) to the final output of the encoder, and we concatenate them as the representation of all sentences which captures more language information.

\[ Com = \sum_{i=1}^{n} \alpha_i s_i \]  

\[ S = [s_i \mid Com] \]

A classifier is stacked on FCRN to calculate the label \( \hat{y}_i \). The classifier is instantiated by a linear layer and a sigmoid function.

\[ \hat{y}_i = \sigma (W_2 S_i + b_i) \]  

IV. EXPERIMENTS

In this section, we present our experiments from text summarization datasets, evaluative criteria, the implementation details of our model, comparison with multiple previous approaches, hierarchy linguistic knowledge analysis and the compute efficiency studies.

A. Datasets

We evaluate our model on two benchmark datasets, namely CNN/DailyMail and XSum. These datasets possess diverse summary characteristics. The proportion of novel n-grams represents the level of abstraction of the dataset. The highly abstractive dataset can reflect the potential of the model to capture the semantic information of text span. We used a greedy algorithm similar to [19] to obtain an oracle summary for each document to train extractive models.

1) CNN/DailyMail: CNN/DailyMail dataset is generated by modifying a question answering dataset, including 93K articles from CNN and 200k articles from Daily mail websites. Following previous work, we conduct experiments on the non-anonymized version and split the dataset into 287,226/13,368/11,490 for training, validation, and testing. Due to the limitations of the memory and the model, the document is truncated to 512 tokens.

2) XSum: XSum dataset consists of 226,711 BBC articles with single-sentence summaries. It has more novel n-grams in the target summaries that do not appear in their source document than CNN/DailyMail dataset. Following Narayan et al. [24], we split XSum into 204,045/11,332/11,334 for training, validation, and testing.

B. ROUGE

ROUGE is a package for the automatic evaluation of system-generated summaries by counting the number of overlapping units between the generated summary and the reference summary. There are three ROUGE metrics extensively used in text summarization, namely the F1 score of ROUGE-1, ROUGE-2, and ROUGE-L. (R1 and R2 are shorthands for ROUGE-1, ROUGE-2; RL is ROUGE-L). R1 and R2 evaluate the summary by counting the overlapping uni-grams and bi-grams, respectively. RL assesses by longest common subsequence. Following previous work, we report R1, R2, RL as a means of assessing fluency.

C. Experimental Setup

1) Implementation details: We implemented our model based on the “albert-based-v2” version of ALBERT [1]. We set the hidden size of LSTM to 384 due to the Layer Normalization. Following Guo et al. [8], the Adam optimizer with a learning rate of \( 2e^{-3} \), \( \beta_1 = 0.9 \), \( \beta_2 = 0.999 \) is used during training. The warmup strategy increases the learning rate from 0 to \( 2e^{-5} \) on the first 15000 steps:

\[ lr = 2e^{-3} \cdot (step^{-0.5} \cdot step \cdot warmup^{-1.5}) \]  

We conduct our experiments on 3 GPUs (RTX 2080 Ti) with 50000 steps. The gradient accumulation with two steps is used to enlarge the batch size. Model checkpoints are evaluated on https://github.com/huggingface/transformers/tree/master/src/transformers/models/albert
TABLE III

<table>
<thead>
<tr>
<th>Model</th>
<th>R1</th>
<th>R2</th>
<th>RL</th>
<th>Speedup_train</th>
<th>Speedup_infer</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCRN-token</td>
<td>43.58</td>
<td>20.48</td>
<td>39.97</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>FCRN</td>
<td>43.66</td>
<td>20.54</td>
<td>40.03</td>
<td>5.9x</td>
<td>1.65x</td>
</tr>
</tbody>
</table>

the validation set per 1000 steps. We saved the three best checkpoints on evaluated losses and record the performance of the best checkpoints on the test set.

We obtained the scores of the candidate sentences and sorted the sentences in descending order based on these scores. Top-3 sentences are selected to form the summary.

2) Trigram Blocking: This is a simple but effective approach to improve performance during inference. Given a selected summary set and a sentence, the sentence will be selected into the summary set when it has no trigram overlapping with the selected summary. We also apply this approach to this work.

D. Experimental Results and Analysis

A comprehensive experiment is conducted on CNN/DailyMail and XSum. We use the official ROUGE script (version 1.5.5) to evaluate the generated summaries. The results are reported as follows.

1) CNN/DailyMail: Table I summarizes our results on the CNN/DailyMail dataset. We list strong baselines with different learning approaches. LEAD-3 is a commonly used and strong extractive baseline. It simply selects the first three sentences as a summary. Rnn-ext+RL [22] applies policy-based reinforcement learning to the extractor. NeUSum [21] extracts the summary by jointly learning to score and select a sentence. BertSumExt [5] extends BERT to text summarization by inserting multiple [CLS], segment information, and stacking a linear layer. TSEM-sentence [8] is the first stage of TSEM to assign relevance scores to sentences. We change it as a baseline by choosing three top-ranked sentences to compose a summary. FCRN\textsubscript{Com} predicts the score of a sentence only by complementary representation Cont. MATCH-ORACLE [7] extracted by BertSumExt is the ground-truth used to train MATCHSUM. This model has achieved the state-of-the-art extractive result on CNN/DailyMail. MATCH-ORACLE-FCRN extracts the ground-truth by our model. MATCHSUM-FCRN is trained on MATCH-ORACLE-FCRN, while MATCHSUM-BERT is trained on MATCH-ORACLE.

From the first block of Table I we observed that our approach outperformed the baseline models. As compared to TSEM-sentence and FCRN\textsubscript{Com}, which only used the final output of ALBERT and the complementary embedding, respectively, our model improved the ROUGE score. It means that the knowledge of the hidden layers as complementary to the final layer indeed achieves better performance. TSEM-sentence performs than FCRN\textsubscript{Com} means semantic information is more critical for text summarization. The fusion representation of hidden layers can weaken semantic information of a sentence.

FCRN provides syntactic and surface information as complementary to semantic features. This further confirms that text summarization requires comprehensive linguistic knowledge.

The second block in Table I presents that the MATCHSUM ground-truth extracted by our model provides a stronger oracle than BertSumExt. The MATCHSUM approach performs better with the MATCH-ORACLE-FCRN.

2) XSum: Our main results on the XSum dataset are shown in Table II Again, we report the performance of ORACLE, LEAD, MATCH-ORACLE-FCRN, MATCH-ORACLE, and BertSumExt where LEAD baseline simply selects the first two sentences from the document. XSum has a lower ROUGE score than CNN/DailyMail because its summary is more abstractive. Following previous work [5], [8], we process the XSum dataset with a greedy algorithm and then conduct experiments on TSEM-sentence. The experiment results demonstrated the effectiveness of ALBERT on the XSum dataset. We can observe that our model with complementary information to the final layer again is superior to all baselines. MATCH-ORACLE-FCRN generates a better ground-truth to train the MATCHSUM.

3) Hierarchy Linguistic Knowledge Analysis: Previous work has proven that surface features are most prominent in lower layers, syntactic features in middle layers, and semantic features in higher layers. In this work, we split all hidden layers of ALBERT into low layers (1-4), middle layers (5-8), high layers (9-12). We take them as the input of FCRN, respectively. It can change the linguistic knowledge complementary to the output of ALBERT. A comprehensive experiment is conducted to detect the effect of different layers on CNN/DailyMail dataset. The results are reported in the third block of Table II.

We observe that the model without middle layers is the worst, and high layers are more effective than low layers. It means that complementary syntactic knowledge improves the overall system performance better than semantic knowledge on CNN/DailyMail dataset. The reason is due to the similarity between the gold and sentences in the generated summary. Meanwhile, the final layer of ALBERT has included semantic information. It makes no difference whether or not the lower layers are added since the lower layers possess the most information about linear words. We also observe the knowledge of all hidden layers as the input to FCRN can achieve better performance.

E. Compute Efficiency

We conduct experiments on CNN/DailyMail dataset to verify the compute efficiency. We trained models with all token embeddings and sentence embeddings, namely FCRN-token and FCRN, respectively. We report the running time, inference time, and the performance of models in Table III FCRN is about 6 times faster than token-level in iterating through the data during training and about 1.65 times during inference. Compared to FCRN-token, FCRN will require significantly less time to train since a sequence of inputs passes through LSTM cell, one at a time. The performance of FCRN-token does not improve with fusion based on token embeddings but,
using sentence embeddings achieves better performance. The above phenomena suggests that using sentence embedding is more compute-efficient and capture the more comprehensive knowledge of each layer. The model performance poorly removing the complementary information, demonstrating the complementary representation is crucial for predicting the summary of a document.

V. CONCLUSION

In this paper, we propose FCRN to showcase how to fuse all linguistic knowledge in hidden layers and enhance the power of ALBERT. Experiments demonstrate constant improvement over baselines on two benchmark datasets. Furthermore, Using sentence embeddings not only achieved better results but also decreased training and inference time. We find that the syntactic information is worthy of note in our model. We plan to consider more syntactic information and import more world knowledge as a part of our future work.

VI. ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (2018YFC0831500), the National Natural Science Foundation of China under Grant No.61972047 and the NSFC-General Technology Basic Research Joint Funds under Grant U1936220.

REFERENCES


[23] Q. Zhou, N. Yang, F. Wei, S. Huang, M. Zhou, and T. Zhao, “Neural document summarization by jointly learning to score and select sen-

BEHIND: a 4W-oriented Method for Event Detection from Twitter

Kang Zeng, Yujia Liu, Xin Song, and Bin Zhou∗
College of Computer, National University of Defense Technology Changsha, China Email: {zengkang, liuyujia, songxin, binzhou}@nudt.edu.cn

Abstract—Event detection from Twitter has attracted attention from researchers in the past decade due to the widespread use of social media. By leveraging the knowledge derived from these events, it is possible to understand what consumers are interested in and give the opportunity for organizations to make better decisions. Numerous studies have proven the advantage of burst detection methods in detecting events in Twitter streams. However, some burst detection methods mainly focus on the bursty characteristics caused by events while the elements in events are not fully utilized. In this paper, we focus on the elements in When, Where, Who, and What (4W) dimensions of events and propose a 4W-oriented event detection method called BEHIND. BEHIND jointly uses Bursty Elements and Heterogeneous Information Network (HIN) for event detection. Bursty Elements are calculated through probability distribution and they are used to select tweets with bursty elements. HIN is used to enhance relevance judgment in 4W dimensions between tweets to help cluster tweets. The tweet clusters are corresponding to events we detected. We used a benchmark dataset to evaluate our method. Experimental results demonstrate that our method achieves higher precision and less duplication rate, and detects more events than the state-of-the-art methods.

Index Terms—data mining, Twitter, event detection, event summarization, 4W

I. INTRODUCTION

Recently, social media has overtaken print media as the main source of news gathering for consumers [1]. In the past decade, automatically detecting events from Twitter has attracted much attention from researchers.

Burst detection methods [2] [3] [4] [5] have been widely studied and applied for Twitter event detection. They focus on detecting events with bursty characteristics (i.e., breaking news). Those methods can help track topics of general interest and detect events in early stages. However, they still introduce some issues that can affect the results of event detection. We present two cases as examples of such issues as follows.

Case 1: There were two events happened at the same time: “Chinese author Mo Yan won 2012 Nobel Prize in Literature” and “The European Union was awarded the 2012 Nobel Prize in Peace”. Both were hotly discussed on Twitter. However, some methods reported an event about “Mo Yan” and “The European Union”, while there was no event happening between them at that time.

Case 2: There were three presidential debates between Barack Obama and Mitt Romney in 2012. Some methods all reported them as “presidential debate”. Those reports could not identify which presidential debate it is.

The first case shows that some burst detection methods confuse two co-occurring and related events, thus reporting an event that is not actually happened at all. The second case shows that these methods cannot discriminate between multiple occurrences of the same type of event.

In order to get more interpretable event detection results, [6] and [7] borrowed the definition of events from journalism. They defined social media events as: a social media event can be represented by When, Where, Who and What (4W) dimensions. This definition can help identify an event by multiple different aspects of information, while it is generic enough to generalize most social media events. Nevertheless, it is still not fully utilized in Twitter event detection. Some methods [7] using this definition simply incorporate features in 4W dimensions into their process.

To better deal with the issues that may be caused by burst detection methods and better use of the 4W representation, we propose a 4W-oriented (When, Where, Who, What) method called BEHIND for event detection. BEHIND jointly uses Bursty Elements and Heterogeneous Information Network (HIN) [8] to Detect events in Twitter stream. We firstly extract elements in 4W dimensions from tweets. Then we select bursty elements of each dimension and use them to filter tweets. This can filter out noisy data in early stages and improve the precision of event detection. Bursty Elements will be discussed in more detail in Section III-A2.

In addition, we build a HIN on the filtered tweets. The nodes in HIN are tweets, and the types of edges in HIN include When, Where, Who and What (4W). We discuss this in Section III-B for more details. We use HIN to help reconstruct the feature representation of tweets to strengthen the connection between tweets based on elements in 4W dimensions. We use Tweet Clustering based on HIN to make it easier to cluster tweets discussing the same event into the same cluster. Tweet Clustering based on HIN will be discussed in Section III-C. This not only reduces repeated reports of the same event, but also increases the possibility of detecting more events. Then we cluster tweets based on the new feature representation. The resulting clusters are the events we detected.

Finally, we concatenate the top elements in 4W dimensions

∗Corresponding author
DOI reference number: 10.18293/SEKE2021-092
of those events to generate event summaries. The summaries can make the detected events more interpretable and discriminative. Because elements in 4W dimensions can jointly verify an event and present more comprehensive information about the event.

To sum up, the contributions of our work are:

- We propose a 4W-oriented (When, Where, Who, What) method called BEHIND to detect events in the irregular text stream of tweets.
- BEHIND jointly uses bursty elements and HIN to detect events. The former is used to effectively select event tweets, while the latter is used to enhance relevance judgment in 4W dimensions between tweets for clustering.
- Comparative experiments show that BEHIND can achieve higher precision and less duplication rate, and can detect more events.
- The events detected by BEHIND are summarized by the top elements in 4W dimensions, which make the detected events more interpretable and discriminative.

II. RELATED WORK

Twitter event detection has been extensively studied over the past decade. For the purpose of better understanding the existing related work, we classify the existing event detection methods based on the common traits they share.

A. Bursty-term-based Methods

Generally speaking, the occurrence of an event always triggers people to discuss it in Twitter. A relevant number of methods detected events by extracting bursty terms from tweets and clustering such terms to get events. Twevent [9] and SEDTWik [3] used tweet segment, which is defined as one or more consecutive words appearing in a tweet to help detecting events. They extracted bursty tweet segments and clustered them to get events. TopicSketch [2] relied on the concept of word acceleration to detect trending topics on Twitter. It calculated the occurrence rate of pairs or triplets of words as the word velocity. The change in velocity within the two time windows is calculated as the acceleration.

However, the textual contents of tweets are sparse and informal, detecting events by bursty terms may detect clusters of terms that are weakly correlated with realistic events.

B. Social-aspect-based Methods

The way people discuss interesting events on Twitter is much more different from the way people share things in their daily lives. Social aspect information can be utilized for event detection. [10] built the relationship between tags to get a graph of related tags and detected bursty tagging events by extracting subgraphs. Generative Latent Dirichlet Allocation Model (MGe-LDA) [11] is a hashtag-based Mutual for detecting events in Twitter. MGe-LDA emphasized the role of hashtags in the semantic representation of the corresponding tweets. MABED [12] is a statistical method that relied solely on the creation frequency of user mentions that users insert into the tweets to detect important events.

Event detection methods focus on the social aspects of Twitter may only detect the most influential events and ignore the small-scale events. Meanwhile, they may require more hyper-parameter, such as the number of top events to detect.

C. Entity-based Methods

Entities are always considered to contain great event information and can help detecting events more efficiently. [13] examined the roles of entities on event detection. They partitioned and clustered documents based on the entities which contained to represent an event. [7] defined semantic categories based on 4W dimensions, which included named entity, mention, location, hashtag, verb, noun and embedded link. They aggregated tweets discussing the same event into one cluster by the similarity measure between those semantic categories. [14] used entities on Twitter Trends to help clustering and used entity clusters to represent events. It addressed scaling issues with new design choices that link event clusters and enable real-time event detection through evolutionary tracing.

Those entity-based methods usually require lots of computational resources and labeled data. furthermore, most of them did not make full use of the textual semantic features of tweets.

III. METHODOLOGY

Fig. 1 shows the architecture of BEHIND. It consists of four components: data processing, tweet HIN building, tweet clustering and event summarization.

A. Data Processing

1) Elements Extraction: We filter tweets by bursty elements in 4W dimensions. This can greatly reduce the computational cost and improve precision of event detection. We use a few advanced NLP tools\(^1\) to extract relevant elements from tweets. We consider “time” extracted from tweets as elements in When dimension, “country” and “location” as elements in Where dimension, “person”, “organization” and “@username” as elements in Who dimension. Generally speaking, elements in What dimension are very diversified. Inspired by the concept of text segment from [9] and [3] which refer to one or more consecutive words, we use text segments to represent the What dimensional elements of Twitter events.

2) Tweets Filtering by Bursty Elements: Thousands of tweets are generated every day, and most of them (i.e., spam, self-promotion, pointless babble) do not contain information to help event detection. Therefore, after extracting event elements, we calculate the bursty elements that may be related to events and discard the remaining ones.

Bursty Element

[9] introduced the concept of bursty segment to detect Twitter events. Bursty segment refers to one or more consecutive words that abnormally burst in tweets within a time window. We only consider the elements in 4W dimensions of events and we only extract the burst elements in 4W dimensions, which

---

1https://github.com/OpenSextant/Xponents
2https://github.com/FraBle/python-sutime
alleviate the misleading effect of useless bursty segments (e. g., thank god, every day).

Let $N_T$ denotes the total number of tweets and $f_{ele,T}$ denotes the number of element $ele$ in time window $T$. $f_{ele,T}$ can be considered as a Binomial distribution $B(N_T, p_{ele})$ and $p_{ele}$ denotes as the expected probability of $ele$ in any random time window. Since $N_T$ is large enough, it can be considered that $E[ele|T] = N_Tp_{ele}$ and $\sigma[ele|T] = \sqrt{N_Tp_{ele}(1 - p_{ele})}$. We use a formula for the bursty probability $P_b(ele, T)$ for $ele$ in time window $T$ defined by [9] as given in (1), where $S(\cdot)$ is the sigmoid function.

$$P_b(ele, T) = S\left(10 \frac{f_{ele,T} - (E[ele|T] + \sigma[ele|T])}{\sigma[ele|T]}\right)$$

(1)

Taking into account the social aspect of Twitter, $src_{ele,T}$ denotes the sum of retweet count of all tweets containing $ele$ in $T$ and $u_{ele,T}$ denotes the number of users who use the $ele$ in $T$. Both of them also affect the precision of event detection.

Finally, we define the bursty weight of the element $ele$ as:

$$w_b(ele, T) = P_b(ele, T)\log(u_{ele,T}) \times \log(src_{ele,T})$$

(2)

We sort the elements by their bursty weights. The top $\sqrt{N_T}$ elements in each dimension are called bursty elements.

After repeatedly comparative experiments, we only keep tweets containing at least two dimensional bursty elements, which can use reasonable computing resources to achieve great event detection results. Besides, $N_f$ denotes as the number of remaining tweets.

**B. Tweet HIN Building**

We use Heterogeneous Information Network (HIN) in BEHIND to enhance relevance judgment between tweets for clustering. Here we introduce some basic definitions based on previous work [8].

**Definition 3.1 Heterogeneous Information Network (HIN)** A Heterogeneous Information Network (HIN) is a graph $G = (V, E)$ with a object mapping function $\Phi : V \rightarrow A$ and a link mapping function $\phi : E \rightarrow R$ while the type of objects $|A| > 1$ or the type of relations $|R| > 1$. $V$ denotes the object set, $A$ denotes the object type set, $E$ denotes the link set and $R$ denotes the link type set.

**Definition 3.2 Meta-schema** Given a HIN $G$, the meta-schema $T_G = (A, R)$ for $G$ is a graph with nodes as object types from $A$ and edges as relations type from $R$.

We show an example of the HIN meta-schema in Fig. 1. The object type in $A$ is tweet and the relation types in $R$ include $4W$(When,Where,Who,What) and Other(e. g., social aspect relations).

**Definition 3.3 Meta-path** Meta-path $P$ is defined on the network schema $T_G = (A, R)$, the specific form is: $A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} \ldots \xrightarrow{R_l} A_l+1$.

The meta-path $P$ defines a combination relationship $R = R_1 \cdot R_2 \cdot R_l$ between node types $A_1$ and $A_l+1$, while $\cdot$ denotes the combination operation between relations.

According to the definitions given above, we use the filtered tweets to build a tweet HIN. We show a few meta-paths instances in Fig. 2. For example, tweet $t_i$ and tweet $t_j$ contain the same element in $4W$ dimensions or contain the same link, these can be used to establish meta-paths between $t_i$ and $t_j$.

**C. Tweet Clustering based on HIN**

After the HIN is built, we reconstruct the feature representation of tweets to better cluster tweets discussing the same event. We introduce pre-trained BERT [15] embeddings as the initial embeddings of tweets. We define the initial embedding of tweet $t_i$ as $h_i$.

1) Feature Aggregation: We reconstruct the embeddings of tweets by feature aggregation in HIN. We use a path-count [16] strategy as the initial similarity measure of two tweets in HIN, which is the number of meta-paths between tweet $i$ and tweet $j$: $e_{i,j} = |p : p \in P|$.

Moreover, if there are meta-paths between two tweets, the two tweets are neighbors of each other. Note that tweet itself is also its own neighbor. The neighbors of $t_i$ is defined as:

$$N_i = \{t_j | e_{i,j} > 0\}$$

(3)
Finally, we aggregate the features from \( N_i \) to tweet \( t_i \) through the normalized similarity measure, so that we get the new embedding \( z_i \) of tweet \( t_i \):

\[
z_i = \sum_{j \in N_i} \frac{e_{i,j}}{\sum_{k \in N_i} e_{i,k}} \cdot h_j
\]  \hspace{1cm} (4)

2) Jarvis-Patrick (JP) Clustering: After getting the new embeddings of tweets, we can get the final similarities between all tweets and all their neighbors by cosine similarities. Then we sort them to get the k-nearest neighbors of each tweet.

Finally, all tweets can be clustered by JP algorithm [17]. In this, we treat all tweets as separate nodes initially, an edge is added between tweet \( t_i \) and tweet \( t_j \) if k-nearest neighbors of \( t_i \) contains \( t_j \) and vice versa. After traversing all nodes, all connected components can be considered as candidate event clusters in time window \( T \), and the remaining nodes without any edges are discarded.

3) Cluster Merging: We extract 4W dimensional elements from tweets in the candidate event clusters. Some candidate clusters without elements in When, Where and Who dimensions are discarded to get better results. To better manage these candidate event clusters, we not only query the event clusters of the current time window \( T \), but also query the event clusters of the time window \( T - 1 \). We compare the elements in When, Where and Who dimensions of each two clusters. The What dimension are not considered here because the number of elements in the What dimension is usually too large. If the elements coincidence rate is greater than 50\%, we merge these two events. The remaining clusters are the events BEHIND finally detected.

<table>
<thead>
<tr>
<th>TABLE I Results of BEHIND and Baselines</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. event</td>
</tr>
<tr>
<td>MABED</td>
</tr>
<tr>
<td>SEDTWik</td>
</tr>
<tr>
<td>BEHIND-noAgg</td>
</tr>
<tr>
<td>BEHIND-tfidf</td>
</tr>
<tr>
<td>BEHIND</td>
</tr>
</tbody>
</table>

D. Event Summarization

Reasonable event summaries can be used to query and manage events. Existing Twitter event detection methods mainly use a keyword set [9] [14] or a representative tweet [18] to describe an event. The former may generate some keyword sets that are not associated with one realistic event. The latter also struggle to find a representative tweet that sums up the whole event due to the brevity of tweets.

We support that a Twitter event may have corresponding 4W dimensional elements. These elements in different dimensions can jointly identify an event, and make events more effectively queried and managed. Therefore, we sort the count of elements in each dimension of those events. Then, we select the top three elements of each dimension for an event and concatenates them to describe the event.

For example, we use “2012-10-17 | us, america, new york | mitt romney, obama | debate, middle class, president obama” to summarize the event of “Second Presidential Debate between Obama and Romney in 2012”. This allows us to observe that there is a debate between Barack Obama and Mitt Romney in New York on October 17, 2012.

IV. Evaluation

A. Dataset and Setup

1) Dataset: To evaluate the performance of BEHIND, we use a huge Twitter dataset called Events2012 [19] to evaluate BEHIND. The entire dataset includes 120 million tweets. Since Events2012 only contains tweet IDs, we use a crawler to get this corpus. Meanwhile, the results of Twitter event detection need to be manually examined, we confirm that it would cost a lot of time to use the entire data set for experiments. For the time constraint and the volume limitation of Twitter, we use the corpus from October 11 to October 17 in Events2012 to evaluate BEHIND in our work.

2) Baselines:
- SEDTWik: SEDTWik [3] is an extension of Twevent [9]. SEDTWik identifies event based on bursty segments and clusters these segments to get the important events. Experiments in [3] have shown that SEDTWik achieves better results than Twevent.
- MABED: MABED [12] is a method of event detection using social aspect feature, which is based on mention anomaly to detect events.
- BEHIND-noAgg: It is a variant of BEHIND, which removes the feature aggregation module and uses the initial embedding generated by BERT for clustering.
- BEHIND-tfidf: It is a variant of BEHIND, which uses TF-IDF instead of BERT pre-training model to generate the initial embedding of tweets. Most of event detection methods use TF-IDF for their clustering module.

3) Experimental Setup: We use three metrics to evaluate results of event detection, which are Number of events (No. events), Precision, Duplicate Event Rate (DERate). all three metrics are referred from [3] and [9].

- No. events: the number of detected events that can be correlated with realistic events.
- Precision: the percentage of detected events that can be correlated with realistic events.
- DERate: the percentage of repeated detected events among all realistic events detected.

For the proposed BEHIND, we remove all retweets from the Twitter stream. Meanwhile, we set a time window \( T \) to be 24 hours, which can be adjusted according to the number of tweets. For the initial embeddings of tweets, we use the BERT model trained by Sentence Transformers [20], which are tuned specifically meaningful sentence embeddings such that sentences with similar meanings are close in vector space. We set the k used in Jarvis-Patrick algorithm as \( N_j/1000 \) to get the best experimental results.
For both SEDTWik and MABED, we use the implementation provided by the authors. The number of top events to be detected is the hyper-parameter of MABED. We set it to 100, which get the best experimental results.

B. Result

1) Event Detection Results: All methods follow experimental setting in IV-A3. Specifically, we used Google News and Wikipedia Page Titles datasets to identify an realistic event. The detailed comparison is shown in table I. From the comparison results, we have the following observations and analyses:

- BEHIND achieves the best performance in No. event and Precision metrics and second-best performance in DERate metric, which shows BEHIND can cluster tweets discussing the same event better.
- MABED has weak performance in the experiments. By analyzing the results of MABED, we find that most of the results are related the event of “Second Presidential Debate between Barack Obama and Mitt Romney” and it always has been one of the most discussed events. Once a hot event occurs, MABED may not be able to detect other smaller-scale events that occur at the same time.
- Despite using the same experimental setup, SEDTWik does not perform as good as in [3]. We suggest that because some tweets cannot be crawled anymore, which affects the results of event detection. [21] also agrees with it, they reported that about 50% of tweet relevance judgments were deleted in Events2012. This also demonstrates BEHIND’s ability to capture relevant judgments.

2) Ablation Experiments: Through the results of variants of BEHIND, we can get the following observations and analyses:

- On the whole, Both BEHIND-noAgg and BEHIND-tfidf perform worse than BEHIND. Nevertheless, they perform overall better than SEDTWik and MABED.
- BEHIND-tfidf achieves the second-best performance in the No. event and DERate metrics, while the result is poor in the Precision metric. We observe some results of BEHIND-tfidf, which aggregates the elements of different events into a cluster. This shows that the initial embedding generated by BERT can be better used to capture semantic relations between tweets in large corpus.
- BEHIND-noAgg achieves poor performance in No. Events metric. This shows that HIN can help capture more relevance judgment between tweets. It brings closer the representation of two tweets that describe different aspects of the same event, though not nearly as similar semantically.

### Table II

<table>
<thead>
<tr>
<th>Event</th>
<th>Event detected by BEHIND, MABED and SEDTWik</th>
</tr>
</thead>
<tbody>
<tr>
<td>New music video by Justin Bieber and Nicki Minaj performing Beauty And A Beat</td>
<td>2012-10</td>
</tr>
<tr>
<td>Red Bull Stratos</td>
<td>NULL</td>
</tr>
<tr>
<td>Second Presidential Debate between Barack Obama and Mitt Romney</td>
<td>2012-10-14</td>
</tr>
<tr>
<td>Cowboys vs Ravens on Oct 14, 2012</td>
<td>2012-10-11</td>
</tr>
<tr>
<td>Hilary Mantel’s novel Bring Up the Bodies won the 2012 Booker Prize for the second time</td>
<td>2012-10-12</td>
</tr>
<tr>
<td>Space Shuttle Endeavour Embarks on L.A. Road Trip</td>
<td>2012-10-11</td>
</tr>
<tr>
<td>Chinese author Mo Yan wins Nobel Prize in Literature</td>
<td>2012-09</td>
</tr>
</tbody>
</table>

### Table I

<table>
<thead>
<tr>
<th>No. event</th>
<th>Event detected by BEHIND, MABED and SEDTWik</th>
</tr>
</thead>
<tbody>
<tr>
<td>2012-10</td>
<td>canada</td>
</tr>
<tr>
<td>2012-10-14</td>
<td>dallas, baltimore, detroit</td>
</tr>
<tr>
<td>2012-10-11</td>
<td>los angeles, wells, fargo</td>
</tr>
<tr>
<td>2012-10-12</td>
<td>null</td>
</tr>
<tr>
<td>2012-10-11</td>
<td>academy of sciences, china</td>
</tr>
<tr>
<td>2012-09</td>
<td>null</td>
</tr>
</tbody>
</table>
3) Event Summarization: We show a sampling of results detected in Table II. The first column is the manually labeled events and the second column is the automatically generated event summaries by BEHIND, MABED and SEDTWik.

Event summaries generated by MABED include main words and common words. MABED assigns weights to those common words. To make the summaries more concise, we remove the weights of them. For each event in SEDTWik, we only take the top ten segments as the event summary. Note that “Not Detected” means that the method does not detect this event, “NULL” in the event summaries generated by BEHIND means that elements of this dimension is not detected.

Next, we discuss two examples in Table II to better demonstrate that our method can get interpretable and discriminative event summaries. The event of “Chinese author Mo Yan wins Nobel Prize in Literature” detected by BEHIND is summarized by “2012-10-11 | academy of sciences, china | mo yan, swedish academy, nobel prize | nobel prize literature, nobel literature prize, christian science”. The summary gives a more holistic picture about an event by elements in 4W dimensions. MABED did not report this event, while SEDTWik only reported a result that contained “Mo Yan”, “EU” and a few irrelevant segments. This corresponds to Case 1 of Section I.

The event of “Second presidential debate between Obama and Romney in 2012” detected by BEHIND is summarized by “2012-10-17 | us, america, new york | mitt romney, obama | debate, middle class, president obama”. SEDTWik does not report this event, while MABED reports it as: “question, debates, romney (president, ask, answering, don’t, amp, answer, obama, mitt)”. The summary in MABED cannot determine which presidential debate it is, but the summary in BEHIND can confirm that this is the second presidential debate held in New York through “2012-10-17” and “new york”. This corresponds to Case 2 of Section I.

From results in Table I and Table II, we argue that using 4W dimensional elements for event detection can both improve detection performance and make detected events more interpretable and distinguishable. While due to the sparsity and irregularity of tweets, we can observe that the elements of some dimensions are missing. But in most cases, the given elements about an event have been able to identify and understand the event.

V. CONCLUSION

Twitter event detection has attracted great interests from both academia and industry. In this paper, we proposed a method called BEHIND for detecting events from Twitter, which mainly included filtering tweets by burst elements and clustering tweets based on HIN to figure out most relevant events. The evaluation based on benchmark dataset shows that BEHIND achieved higher precision and less duplication rate, and detects more events than the state-of-the-art methods. Meanwhile, BEHIND can derive interpretable event summaries. For potential future work, we consider using natural language generation to improve the readability of event summaries.

ACKNOWLEDGMENT

This work is supported by the National Key R&D Program of China under Grants (No.2018YFC0831703).

REFERENCES


ATFE: A Two-dimensional Feature Encoding-based Sentence-level Attention Model for Distant Supervised Relation Extraction

Shiyang Li¹, Qianqian Ren*¹, Zechao Liu²
¹Department of Computer Science and Technology, Heilongjiang University, Harbin 150080, China
²School of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China
*Email: renqianqian@hlju.edu.cn

Abstract—Distant supervised relation extraction has recently attracted researchers attention in the knowledge graph. However, the current feature encoding model of sentences can not fully represent the features in sentences, which poses a challenge. To solve this problem, we propose a two-dimensional feature encoding-based sentence-level attention model for relation extraction. In this model, we first employ bidirectional long short-term memory networks (BiLSTM) to capture the temporal dependency of the words in the sentence. Then we employ multi-dilated convolution to obtain the higher-level semantic units hidden in the sentence. Afterwards, we combine the above two-dimensional features to embed the encoding of sentences, which is expected to enhance the model’s ability to express sentence features. Finally we build sentence-level attention to complete the relation extraction task. Compared with other excellent methods, the proposed approach provides a significant performance improvement.

Index Terms—Deep learning, Attention, Distant supervision, Relation extraction

I. INTRODUCTION

In recent years, the growing commercial interest in artificial intelligence related fields has spurred the development of knowledge graphs. Many knowledge graphs related techniques have been proposed and applied. Among them, the knowledge base widely used for natural language processing (e.g. Freebase, Wikidata) related issues such as search engines and question answering systems. Some existing large-scale knowledge bases are composed of a large number of triples (e.g. <Jack_Ma, founder, Alibaba>), which implement information storage in a structured manner etc[2, 4]. These triples concisely reflect the two objective entities and the relation between them. However, these existing knowledge bases are not sufficient to cover all the facts in the real world. We need to continuously expand our knowledge base to increase its integrity. Many researches focus on the study of relation extraction, which can automatically obtain unknown relations in knowledge bases from plain text.

Relation extraction is the process of obtaining two entities and the relation between them from an unknown text. This is meaningful during the expansion of the knowledge base. The initial extraction is performed in a supervised manner, requiring people to manually label the training data, which is a time-consuming and very expensive task[3, 5]. Until 2009, the concept of distant supervision is proposed. This method automatically generates relational training data by aligning entities in the text with the known triples in the knowledge base. Distant supervision can effectively avoid the tedious and time-consuming manual annotation process [13]. However, this method can also mislabel and generate a lot of noise during the experiment. To solve the problems caused by distant supervision, multi-instance learning is proposed and widely used [6]. Lin et al propose a sentence-level attention method to make full use of the relation information in all sentences [10]. In this method, the weighted sum of all sentences in the package is used to express the relation between entity pairs. Guo et al add entity recognition on the above method to further obtain entity background knowledge to improve relation extraction performance [9].

In recent years, various deep learning architectures have been proposed to replace traditional natural language to encode sentence features. Convolutional neural networks (CNN) are proposed to code sentence semantic [1, 11, 14, 18]. On this basis, deep convolutional neural networks (DCNN) and residual networks are proposed to enhance the model’s ability to express sentence features [7]. Recurrent neural networks (RNN) and long short-term memory networks (LSTM) are generally adopted to model the temporal dependency of words in the sentence and achieved certain results [19, 20]. These techniques along with other tricks are usually combined to improve the effectiveness of models.

In this paper, we propose a two-dimensional feature encoding-based sentence-level attention model (ATFE) for distant supervised relation extraction. We first build a two-dimensional feature encoder to embed the encoding of sentences, which is expected to combine the temporal dependency of the words and the higher-level semantic units in sentences to enhance the model’s ability to express sentence features. Afterwards, we use the result of two-dimensional feature encoding to build sentence-level attention to complete the relation extraction task. Experiments on real data sets show that compared with baseline models, our model can perform sentence feature encoding more precisely, moreover further improve the performance of relation extraction related algorithms.

The contributions of the paper are as follows:
• We propose ATFE, a new sentence feature encoding model to obtain the two-dimensional feature representation of the sentence.

• We incorporate the sentence-level attention mechanism with our model to calculate the extracted relation probability.

• We implement experiments on real data sets to validate the performance of our proposed model. The experimental results show that our model can encode sentences feature better. It can be more effectively used in the distant supervision relation extraction model to improve task accuracy.

II. RELATED WORK

Most existing relation extraction methods can be roughly divided into two categories: one is based on word sequence, and the other is based on the dependency tree. The method based on the dependency tree is to model the dependency tree of the sentence instance as the input data, which will not be described in detail here. The method based on the word sequence is to use the word sequence to build a model. The model is used to encode sentence features to obtain the semantic representation of sentences. Since deep learning was proposed, neural network models have brought tremendous changes to the research on feature extraction. Nowadays, using neural networks to automatically learn features in sentences for relation extraction tasks has been widely studied. Some classic models for feature encoding of sentences have been proposed, such as the piecewise convolutional neural network(PCNN) model [17]. The researchers applied piecewise max-pooling to the model to make it better encode sentence structure information.

Although the method effectively improves the effect of the relation extraction task. However, due to the influence of the CNN network structure, the small size convolution kernel cannot capture the temporal dependency of the words in the sentence and perform more accurate sentence feature coding. In subsequent works, the researchers introduced RNN to the task of relation extraction and obtained long-term dependency by capturing the time sequence information of the words in the sentence. Among them, long short-term memory networks(LSTM) is an excellent RNN model that is composed of computing units.

These works actively promote the improvement of the accuracy of the relation extraction model, and achieve great success. However, they all ignore the higher-level semantic unit information in the sentence. The semantic unit is shown in figure 1:

Different from word-level information, it is higher-level information hidden in phrases or sentences. They are combined to express the semantics of sentences. Therefore, we believe that accurately capturing the representation of semantic units in a sentence is the key to enhance the model’s ability to express sentence features.

Encoding based on semantic information in sentences has attracted many researchers in natural language processing(nlp) recently. The diversity of models based on deep learning enables them to try different methods to improve the effectiveness of various nlp tasks, such as semantic segmentation, text emotion analysis and machine translation. To the best of our knowledge, this is the first effort to adopt the two-dimensional feature encoding model including BiLSTM and multi-dilated convolution in distant supervised relation extraction.

III. METHODOLOGY

For $n$ sentences $\{x_1, x_2, x_3, \ldots, x_n\}$, each sentence consists of $m$ words, denoted as $x_i = \{a_1, a_2, a_3, \ldots, a_m\}$, which contains two entities (head_entity and tail_entity). The purpose of our model is to calculate the probability of each relation $r$. For the entire relation extraction model, we divide it into two parts:

- **Two-dimensional feature encoder:** Given $n$ sentences $\{x_1, x_2, x_3, \ldots, x_n\}$, use our proposed model to perform feature encoding on sentence vectors.
- **Sentence-level attention:** We make full use of the multi-instance learning idea, extract sentence information of the target entity pair through all contained relation to predicting the relation probability of the target entity pair.

A. Two-dimensional Feature Encoder

The structure of the two-dimensional feature encoder is shown in figure 2. The model is composed of vector representation, network layer, piecewise max pooling. The following describes how the model is implemented:

**Vector representation.** Since the neural networks cannot directly recognize the words in the sentence, we should use the encoding tool to transform the words into low-dimensional vectors. Considering that the length of each sentence is different, and important information may be contained anywhere in the sentence, we pad zeros around sentences to make them equal in length, which is in order to facilitate the model encode the sentence vector. And we add the position information of the given entity pair.

**Word embedding and position embedding.** Word embedding is distributed representations of words. It can map words in texts to a low-dimensional vector that can capture syntactic and semantic meanings. Position embedding is an important part of the model. It is defined as the combination of the relative distances from each word in a sentence to two given entities, as shown in figure 3. The final embedding method is shown in Vector Representation in figure 2. If the specified
word embedding is $d_w$ and the position embedding is $d_p$, given the vector sequence $x_i = \{a_1, a_2, a_3, \ldots, a_m\}$, the length of $a_i$, denoted as $da_i = d_w + 2 \times dp$. In our model, we set $d_w = 5$ and $d_p = 1$, then $da_i = 5 + 2 \times 1 = 7$.

![Fig. 2. Two-dimensional feature representation sentence encoder.](image)

![Fig. 3. The distance from "co-founder" to the head entity "Jack Ma" in the sentence is 4, and the distance to the tail entity "Alibaba" is 2.](image)

**Network layer.** The matrix containing sentence word embedding and position embedding is input into the network for feature encoding. First, BiLSTM is used to capture the timing information of the words in the sentence to obtain long-term dependencies. The essence of BiLSTM is a two-way LSTM structure.

**Temporal dependency capturing.** LSTM is a kind of recurrent neural network and has been widely used. By adding control gates (including input gates, output gates and forget gates) to the network, the network can eliminate unnecessary words in sentences and retain important words. However, due to the structural characteristics of LSTM, it is impossible to encode the information from back to front. It makes the network unable to carry out more fine-grained encoding. To solve this problem, BiLSTM first performs a LSTM from front to back, and then performs a LSTM from back to front, and next combines two results to obtain the final feature encoding. In the model, we input sentences into the network in word order and use the characteristics of the words that enter the network first, which is calculated together with the next word that enters the network. This process is repeated until the last word is processed. The process is shown in figure 4.

**Semantic information capturing.** Based on the representation generated by BiLSTM, we introduce multi-dilated convolution to capture the semantic unit representation in the sentence or phrase through the temporal dependency in the sentence. Dilated convolution is actually a special CNN design. By adding "holes" to the convolution, the receptive field can be expanded exponentially without adding additional parameters. To prevent the dilated segments of the convolutional kernel from causing the missing of vital local correlation, we design the network as a two-layer convolution with different dilation rates, as shown in figure 5.

![Fig. 4. First we input "Jack_Ma", "is", "the", "earliest", "co-founder", "of", "Alibaba" in turn to get three hidden vectors and then input "Alibaba", "of", "co-founder", "earliest", "the", "is", "Jack_Ma" in turn to get three hidden vectors. Finally, the hidden vectors are spliced to get the encoding of the sentence.](image)

![Fig. 5. Multi-dilated convolution structure.](image)

In the proposed model, the scale of the convolution kernel is 3, and the dilation of the two-layer dilated convolution is 1 and 2, respectively. In this way, each convolution kernel in the highest layer can observe 7 inputs from BiLSTM from left to right. While expanding the receptive field, it also prevents the highest level from processing sentence information that is too long, and reduces the noise in the sentence caused by the influence of some irrelevant information. At the same time, to remain the sentence vector sequence dimension of the input and output of each network layer, we set the different padding size to make the same length of the convolution process. In this way, we can capture the semantic unit representation from phrase-level information with a smaller dilation rate and sentence-level information with a larger dilation rate.

**Two-dimensional feature representation.** Finally, we will perform one-dimensional splicing of the results obtained by two-layer convolution to obtain feature vectors containing sentence timing information and semantic units. The feature vector is passed through a convolutional layer with a convolution kernel 1 to output the two-dimensional features of the final sentence.

The convolution process with convolution kernel 1 is equivalent to the calculation process of the full connection. It can...
increase the nonlinearity of the network by changing the vector dimension, thus the network can express more complex features.

**Piecewise max pooling.** According to the given entity pair, the sentence is divided into three parts \((p_1, p_2, p_3)\), and then the maximum pooling operation is performed on each part (the piecewise max pooling part is in figure 3, the gray block represents the position of the entity pair in the sentence). The process can be defined as:

\[
[x]_{ij} = \max(p_1), \max(p_2), \max(p_3)
\]  

(1)

**B. Sentence-level Attention**

To make full use of the relation information expressed in each sentence in multi-instance learning, we use sentence-level attention to complete the relation extraction task[8]. Its structure is shown in the figure 6:

![Fig. 6. Sentence-level Attention.](image)

Given a set of entity pairs \(<e_h, e_t>\), the set \(S\) consists of all sentences containing this entity pair, denoted as \(S = \{x_1, x_2, x_3, ..., x_n\}\). We perform weighted summation on the feature encodes of all sentences in the set to get the feature encode of set \(S\):

\[
s = \sum_{i} w_i x_i
\]

(2)

To measure the contribution of the sentence containing the entity pairs in predicting whether the entity pair has a relation \(r\), we first match the vector representation of the relation \(r\) with \(x_i\), we calculate their similarity:

\[
f_i = \|x_i - r\|_2
\]

(3)

Here, due to the nature of word embedding, we use entity vector difference to represent the relation characteristics between entity pairs[12, 16]:

\[
r = e_h - e_t
\]

(4)

Then \(f_i\) represents the score using the sentence to predict whether the entity pair has a relation \(r\), and we calculate \(w_i\) according to the following formula:

\[
w_i = \frac{\exp(f_i)}{\sum_{k} \exp(f_k)}
\]

(5)

In this way, we get a vector representation of the set \(s\) of sentences containing a given entity pair. We use a linear function to represent the score of each possible relation \(r\), it is the final output of the neural network.

\[
O = RS + d
\]

(6)

Where \(R\) is the representation matrix of the relation, and \(d\) is the deviation vector. Then we get the scores for \(i\) possible relations. We calculate the probability of each relation \(r\) accordingly:

\[
p(r | s, \theta) = \frac{\exp(o_r)}{\sum_{k=1}^{n_r} \exp(o_k)}
\]

(7)

Where \(n_r\) is the total number of distant supervised alignments.

**IV. TRAINING AND EXPERIMENT**

This section introduces the optimization process, data set, experimental environment, and parameter setting of our model respectively. Finally, the performance of the model is compared with some baseline methods.

**A. Model Optimization and Training**

For the entire relation extraction model, we use cross entropy to define the objective function as:

\[
J(\theta) = \sum_{i=1}^{s} \log_p(r_i | s_i, \theta)
\]

(8)

We use Adam to optimize the objective function. Adam has a great advantage in non-convex optimization problems and is an extension of Stochastic Gradient Descent (SGD). It uses the first-order moment estimation and the second-order moment estimation of the gradient to dynamically adjust the learning rate of each parameter. It can maintain high-efficiency calculations while occupying a small amount of computer memory. When training the model, we introduce dropout to prevent the model from overfitting to get the best model training results[15].

**B. Datasets and Preprocessing**

Here we use Nyt10 data set for the relation extraction task. The word2vec tool is used for sentence vectorization.

**Nyt10.** The nyt10 data set is maintained by Tsinghua University and is used for relation extraction experiments. It is generated by adjusting the New York Times Corpus and Freebase. A total of 53 kinds of relations are included, among which there is a relation NA, which indicates that the head entity and the tail entity in the sentence have no relation. We use the nyt10 data set for model training and verification. The training set includes 466876 sentence examples. The test set consists of 172448 sentence examples.

**Parameter settings.** In order to determine the optimal settings of each parameter of the model, based on the data provided by [10, 11, 18], we made more attempts on the sentence embedding size, window size, learning rate, and other parameters. In the process of multi-dilated convolution, try the window size and the number of convolution layers between {2,
The result is shown in figure 8. It can be seen from the figure that due to the limitations of the network structure, the final results of the ABiLSTM and ADNN models are not satisfactory, and the two-dimensional feature representation model (ATFE) that combines sentence time information and higher-level semantic representations produce a satisfactory result.

Furthermore, the AUC and F1 values of our proposed ATFE model in the validation set and test set are further compared with the APCNN model. The result is shown in figure 9:

Through the above evaluation, we can find that the performance of our proposed ATFE model on the test set and verification set is significantly higher than the APCNN model. In the validation set, the highest AUC value of the ATFE model can reach 0.6404, and the highest F1 value can reach 0.6213. The APCNN model only has an AUC value of 0.5952 and an F1 value of 0.5686. Similarly in the test set, the final AUC and F1 values of the ATFE model are 0.3642 and 0.3443, respectively. The AUC and F1 values of APCNN are 0.3162 and 0.3078, which are lower than the ATFE model.

The above several experiments show that our model has better performance than other baseline models. It shows that capturing higher-level semantic representations through the temporal dependency of words in sentences, obtain the two-dimensional feature encoding of the sentence can improve the model’s ability to encode sentence features, and accordingly enhance the performance of relation extraction tasks based on remote supervision. It played a positive role in promoting the development of this research work.
Experiment summary and questions. It can be seen from figure 7 that when the recall is low (less than 0.05), the precision of our model has a rapid decline. It is because the heldout evaluation suffers from false negative in Freebase. Although this problem can be eliminated by manual evaluation, it will inevitably lead to huge labor costs as the size of the data set increases. So whether or this false negative label can be eliminated or corrected through the autonomous learning of the model will become a direction for future researchers to improve the performance of the model.

V. Conclusion

In this paper, we propose an ATFE model to solve the problem that the feature encoding model of sentences in the previous distant supervised relation extraction method cannot adequately represent the features in sentences. We first use BiLSTM to capture the time dependence of words in sentences by bidirectional encoding. On this basis, we design multi-dilated convolution to further acquire the higher-level semantic units hidden in sentences. Finally, the feature encoding ability of the model is maximized by fusing the captured two-dimensional features, We build sentence-level attention to complete the relation extraction task. Compared with similar methods, our proposed method has significant performance improvement.

Acknowledgement

This work was supported by the Basic Research Program (no. JCKY2019604B002).

References

Extracting information from driving data using k-means clustering

Nour Chetouane† Lorenz Klampfl and Franz Wotawa
Christian-Doppler Laboratory for Quality Assurance Methodologies for Autonomous
Cyber-Physical Systems, Institute for Software Technology, Graz University of Technology
Inffeldgasse 16b/2, Graz, Austria
{nour.chetouane, lklampfl, wotawa}@ist.tugraz.at

Abstract

There is an increasing availability of data, but for making decisions and other tasks we need information. Hence, we require to analyze the data and extract parts or come up with relations between different pieces. In this paper, we focus on information extraction within the automotive industry. In particular, we report on applying k-means clustering for identifying episodes in vehicle data. An episode is considered to be a time interval where a vehicle is performing an activity worth being distinguished. The underlying idea is to cluster the data such that we are able to extract such similar situations like breaking before a crossing only considering vehicle data. We discuss a method that allows extracting such episodes capturing actuator and sensor readings over time. Besides introducing the underlying method, we present obtained empirical results making use of a freely available dataset showing that the extracted episodes have indeed a meaningful interpretation.

1 Introduction

We live in a world of increasing availability of data. However, for obtaining information, i.e., data with uncertainty of interpretation removed, required to fulfill certain tasks, we need to analyze available data and set it in relation to a context. This may also lead to removing redundancies as well as coming up with relations between different pieces of data worth being considered in a given application context. In this paper, we focus on the automotive domain. Current vehicles produce a lot of data obtained during driving. Data include vehicle speed, breaking pressure, or the position of the steering wheel, and can be obtained when monitoring the respective bus systems of a car. When driving, however, we see a limited amount of situations worth being distinguished. This includes braking before a crossing or accelerating after stopping. The question now is, whether we are able to “see” such distinguishable situations also in vehicle data.

In order to answer the question, we propose an approach utilizing clustering for obtaining time intervals we call episodes, and to evaluate whether those episodes can be assigned a meaningful interpretation. The underlying idea behind the approach can be summarized using the overall considered data analysis process depicted in Figure 1. We start with time series data and apply clustering. Ideally, the clusters comprise data points that are falling within a certain time interval. In a second step, we are considering time episodes for clusters and select one of these as representative.

In order to show that the approach really work in practice, we carried out an experimental evaluation relying on the freely available dataset from Audi [2]. This dataset comprises vehicle data but also images from attached cameras allowing us to interpret obtained episodes. Besides a detailed description of the evaluation, we discuss the obtained results.

Applications of our approach in the automotive industry include extracting episodes for testing and in particular test case generation. We can use the episodes in two different ways. First, we make use of episodes for concretizing abstract test cases. An abstract test case state a sequence of actions like accelerating, braking, turning left or right, or driving constant speed. The episodes themselves allow to concretize those abstract actions considering the concrete values for acceleration, braking, etc. Second, the episodes provide means for basic behavior that shall be considered in testing. The extracted episodes in a more abstract meaning provide situations that occur during driving. Hence, we may use these episodes as basic actions for generating arbitrary sequences of actions to be executed for testing.

*The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.
†Authors are listed in alphabetical order.
DOI reference number: 10.18293/SEKE2021-118
The paper is organized as follows. In the next section, we discuss and formalize our episode extraction approach in detail. Afterwards, we introduce the processes carried out for evaluating the approach and present the obtained evaluation results. Finally, we conclude the paper and discuss future research.

2 Information extraction using k-means

In the following, we outline our clustering and episode extraction algorithm in detail. We start discussing the data and requirements on them. For the approach, we assume a set of data $D$ over time provided for given attributes $a_1, \ldots, a_n$. We assume $D$ is of the form $\{v_{t_0}, \ldots, v_{t_m}\}$ where $v_t$ is a tuple $(v_1, \ldots, v_n)$ at time point $t$ storing values of their corresponding attributes. We, furthermore, assume that the given dataset is already cleansed and attribute values are available for all points in time. In addition, and for simplification, we assume that the delta time between $t_i$ and $t_{i+1}$ is $\Delta t > 0$ for all $i$ from $0$ to $m - 1$.

It is worth noting that in practice data may not fulfill these requirements and need to be cleansed and modified. For example, vehicle data often does not follow the requirements regarding time. There may be no centralized clock, which would be necessary to assure that values are captured at specific points in time. Hence, we need procedures for mapping the original data to the form that is required. This may include making approximations or assumptions, e.g., using splines for interpolating values or assuming that values do not change unless otherwise stated. For vehicle data, these modifications seem not to be a problem, because of the frequency used to obtained sensor data.

In this study we propose an approach mainly based on clustering analysis, the general principle behind clustering is to maximize the similarity between elements of the same cluster and to also maximize the dissimilarity of elements from different clusters. The main advantage of clustering analysis is that it is simple, easy to use and has been shown effective for serving several machine learning and data mining purposes. It mainly consists in splitting a set of unlabeled data into a fixed number of clusters. K-means clustering works as follows: First, $k$ initial centroids are randomly chosen from the original dataset. At each iteration, the algorithm goes through the data points and computes the distance between each point and the $k$ centroids. The distance can be computed using for example Euclidian distance for numerical data or other types of distance measures depending on the type of data to be clustered. Each data point is then assigned to the cluster having the nearest centroid. After partitioning all data points, each centroid is re-calculated as the mathematical mean of each cluster, i.e. the sum of all the data points belonging to that cluster divided by the number of elements in the group. The process of data points partitioning and centroid adjustment is repeated until each centroid value is stable giving the final clustering of the input data.

Using k-means and given a certain value for the number of clusters $k$ to be computed, we obtain the clusters $C_1, \ldots, C_k$ where $i \in \{1, \ldots, k\} : C_i \subseteq D$. Note that all clusters are distinct, i.e., for all $i, j \in \{1, \ldots, k\}$ where $i \neq j$: $C_i \cap C_j = \emptyset$. It is worth noting that we are not considering time as an attribute when clustering. Furthermore, clusters may provide partitions over time that are not connected. We assume to points of data $v_t$ and $v_{t'}$ from $D$ to be connected if and only if $|t - t'| = \Delta t$. We call these two points approximately connected if $|t - t'| \leq m \cdot \Delta t$ for any integer value $m$. A subset of a dataset is said to be (approximately) connected if all data points in there are
(approximately) connected. In the second step we select a cluster $C_i$ and extract a connected subset. A connected subset of a cluster $C_i$ can be formally defined as follows: $C_i^c \subseteq C_i$ such that $\forall v_t \in C_i^c \rightarrow \exists v_{t'} \in C_i^c : |t - t'| \leq m \cdot \Delta t$

Note that such a subset $C_i^c$ may not comprise all data points of the original set $D$ between the minimum time and the maximum time of $C_i^c$. Hence, we need to complete such a set using missing data points from $D$ resulting in an episode of Cluster $C_i$. An episode of an approximately connected subset is a set comprising all elements of the subset and all elements of the original dataset $D$ that fall within the time interval of the subset but have not been considered. Formally we define a function $E$ on approximately connected subset returning an episode as follows: $E(C_i^c) = C_i^c \cup \{v_t | \exists v_{t'}, v_{t''} \in C_i^c, t' < t < t'' : v_t \in D \setminus C_i^c\}$

In this case, we also write $E_i$ for referring to an episode that belongs to the connected subset $C_i^c$, i.e., $E_i = E(C_i^c)$.

The algorithm $\text{EE}$ summarizes the discussion on how episodes for a given dataset are computed:

\begin{algorithm}
\caption{$\text{EE}(D_I, m, k)$}
\begin{algorithmic}[1]
  \State Let $\text{Sols}$ be $\{\}$
  \State Let $D$ be the cleansed and modified set of data originating from $D_I$
  \State Let $C_1$ to $C_k$ be the $k$ clusters obtained calling $k$-means.
  \For {$i = 0$ \text{to} $k$}
    \State Let $C_i^c$ be one approximately connected subset of cluster $C_i$ considering the parameter $m$.
    \State Let $E_i$ be $E(C_i^c)$.
    \State Add $E_i$ to $\text{Sols}$.
  \EndFor
  \State Return $\text{Sols}$
\end{algorithmic}
\end{algorithm}

Algorithm $\text{EE}$ obviously terminates. Its computational complexity is determined by $k$-means clustering. Hence, in the worst case the runtime is exponential.

3 Experimental evaluation

The objective behind the experimental evaluation outlined in this section is to show whether $k$-means clustering works on real world driving data and allows deriving distinguished driving scenarios having a meaningful interpretation, like braking before stopping in front of a crossing. In the following, we discuss the setup of the evaluation and results obtained.

Setup: In order to carry out the experiments, we make use of the public available Audi Autonomous Driving Dataset (A2D2) \cite{figueroa2022a2d2}. It includes images and 3D point clouds, semantic segmentation, instance segmentation, plus automotive bus data. In this study, we focus on the vehicle bus data which corresponds to three different driving scenarios recorded in three cities in Germany: Gaimersheim, Ingolstadt and Munich. The data comprises 22 attributes with corresponding timestamps and units. Several sensors are used to measure for example; acceleration pedal, (angular) velocity, GPS coordinates, brake pressure, pitch and roll angles, steering angle, vehicle speed, etc. A2D2 dataset also includes sequential camera images corresponding to each city, we have made use of the camera front images in a second step of our experiment in order to map clustered episodes to sequences of videos and check whether the clustering is capable of finding similar scenarios.

The approach is implemented in Python 3 and for running the $k$-means clustering algorithm, we make use of python-weka-wrapper\footnote{see https://pypi.org/project/python-weka-wrapper3/} package which runs different machine learning algorithms from the open source library WEKA\footnote{see https://www.cs.waikato.ac.nz/ml/weka/}. For carrying out the experimental evaluation we use a MacBook Pro (2017) with a 2.8 GHz Intel Core i7 processor running under Mac OS High Sierra Version 10.13.

Before conducting the clustering, we first perform a data pre-processing step. As in the original dataset, each attribute $a$ (sensor) values were recorded in a different time axis. To carry out clustering on data points, we performed data interpolation using same time axis for all attributes. Therefore, we looked for the minimum and maximum recorded timestamps for all sensors, then, created a common time line for all attributes by setting $t_0$ as the minimum recorded timestamp and continue to add the smallest time difference $|t - t'|$ between all recorded timestamps of all the data sensors, until reaching the maximum timestamp recorded in the data. To make the data interpolation, we used a Cubic Spline function which calculates an interpolating polynomial that has small error. The interpolation simulates each function corresponding to an attribute $a$ with the original values recorded at an initial different $\Delta t$, to be used afterwards to compute new data points given as input the new created timeline for all the data attributes. For mapping bus signals to corresponding camera images, we have also performed an interpolation on images timestamps to synchronize them with the bus signals. Further on, in order to achieve clearer interpretation and obtain more precise results, we have also carried out data cleansing where all values of the brake pressure attribute which are $< 0.2$ were set to 0.

Results: During experiments, we focused on four attributes: acceleration pedal [%], brake pressure [bar], steering angle [°], and vehicle speed [km/h]. We run experiments with different values of $k$. After several trials, we noticed that clustering with $k = 6$ yielded to a better separation of clusters. We have also made our choice by evaluating the similarity between the obtained episodes in every cluster. For this, we computed the Pearson correlation co-
efficient to measure the strength of the linear relationship between each pair of episodes. In the following, we report the results for an example of the obtained clusters when performing k-means with \( k = 6 \) on the Gaimersheim dataset. Figure 2 represents the obtained driving episodes in cluster 5 when performing k-means with \( k = 6 \). It shows, for each of the four selected attributes, graphs of the data points which get clustered in cluster 5. This cluster includes highly similar episodes and represents a turning maneuver. As we can notice in Figure 2, most episodes are showing pedal acceleration values and vehicle speed progressively increasing, basically no braking is occurring, and a parabolic curve corresponding to the steering angle showing episodes values increasing from \( 100^\circ \) to approximately \( 350^\circ \) and decreasing back to around \( 150^\circ \) which highly indicates a turning maneuver.

We computed Pearson coefficient between each pair of episodes as the covariance of the two episodes values divided by the product of the standard deviation of each episode. A score close to 1 shows a large positive correlation, whereas a score close to -1, indicates a large negative correlation and equal to 0 means no significant correlation exists between the two variables. Figure 3 presents four heatmaps, each corresponds to one attribute. The color red indicates a high Pearson correlation coefficient between majority of the episodes, for the attributes: acceleration, steering angle and vehicle speed. For the braking pressure attribute, some of the coefficient values obtained show high correlation, noting that several correlation scores were not computed since the Pearson coefficient cannot be measured if one of the variables is 0 which is the case for braking pressure attribute as all values \( \leq 0 \) were set to 0 in the data cleansing step. When performing clustering using the Gaimersheim dataset example, we mainly noticed high correlation for the vehicle speed attribute mostly for all clusters which results of the fact that k-means is mostly using vehicle speed as the dominant attribute in clustering the data. This also can be explained by the fact that there is more variation in vehicle speed values in the used dataset than the other selected attributes values (see Figure 4). We have also measured the probability density distribution for episodes in each cluster. Figure 5 shows four histogram plots corresponding to each attribute in cluster 5 for \( k = 6 \). We can see that the majority of episodes pedal acceleration values are more or less similarly distributed, as they are mostly arranged between \( 0\% \) and \( 30\% \). Regarding the speed, the majority of episodes have a maximum value reaching \( 20km/h \). Vehicle speed values in this cluster are slowly elevating from \( 5km/h \) to around \( 20km/h \). Similarly with the steering angle, as the values are equally spread as they are increasing from \( 100^\circ \) and the majority of episodes values exceed \( 250^\circ \), and some of them even reach \( 350^\circ \) as shown in Figure 2. For the brake pressure, it is 0 for all episodes.

In Table 1 we report and interpret information extracted from each cluster for the three dataset examples. Based on the changes in the graphs of episodes for every attribute, we could see that k-means clustering could actually separate, to a certain level, different driving situations in different clusters and we were able to observe and interpret similar driving scenarios represented by the episodes in the same cluster. It is worth noting that we further verified our interpretations by mapping episodes to sequences of videos created using corresponding front camera images. When observing clustered episodes, we could distinguish several driving scenarios, for example, an increase of vehicle speed from 0 to
Figure 4: Frequency distribution of attributes: acceleration pedal, vehicle speed, braking pressure, and steering angle in the original dataset in the Gaimersheim example.

Figure 5: Probability Density Distribution of episodes for attributes: acceleration pedal, vehicle speed, braking pressure, and steering angle, in cluster number 5 in the Gaimersheim example.

A certain value indicates that the vehicle was stopping and starting back again, a high decrease in vehicle speed that reaches 0 along with a brake pressure occurring shows that the car is braking before stopping in a crossroad or a traffic light, also a progressive decrease of vehicle speed and acceleration pedal indicates that the vehicle is approaching an obstacle or a crossroad. A high decreasing or increasing in steering angle refers to the vehicle turning.

For the Munich and Ingolstadt dataset examples, we have noticed some clusters describing similar driving scenarios as these two datasets were recorded in a high traffic environment, for example the vehicle braking when approaching an obstacle like a pedestrian crossing the street or another car during traffic jam. We also found new driving situations for instance a car completely stopping at a red traffic light (cluster 0 in Munich dataset) which didn’t occur in the first Gaimersheim example as this one was recorded in low traffic, noting that Gaimersheim example does not include crossroads with traffic lights.

Nevertheless, some similarities were seen between cluster 0 and cluster 2 in the Ingolstadt example which led to few misclassifications. Cluster 1 represents the vehicle driving after making a turn in a crossroad as it shows a sudden decrease in steering angle values while in cluster 2, episodes show the vehicle after crossing or stopping in a traffic light with no turning occurring however we could see two episodes from cluster 2 showing the car driving after making a turn. Also, another limit, is that we couldn’t differentiate for instance in cluster 1 in Munich example and cluster 2 in Ingolstadt example, whether the car was stopping in a traffic light or in a traffic jam or because of an obstacle. Yet, we plan to tackle this issue in future work by adding additional information to the clustering algorithm with the use of an object detector.

In summary, we state the k-means clustering was to a certain level able to group together similar driving scenarios. Some clusters included highly correlated episodes such as the ones representing turning, stopping and braking before stopping. We were also able to analyze the clusters results based on the changes in the values of each attribute and come up with reasonable interpretations using the available camera images.

Threads of validity: Regarding internal threads we have to say that the analysis regarding the interpretation of clusters was done manually. Hence, the reported results are to an extent subjective. However, two of the authors carried out the analysis separately to mitigate this thread. Furthermore, we did data cleansing and modifications before carrying out the study to assure that the data meets the requirements. We believe that these changes as described in this paper are reasonable and should not influence the outcome of the evaluation. External threads include the use of a particular dataset and, hence, generalizability of conclusions may be in question. However, the dataset comprises at least different driving routes and situations. Nevertheless, further studies also including different application areas are required.

4 Conclusions

In this paper we introduced an approach for extracting information from real world driving data based on k-means clustering. We mainly try to answer the question whether k-means clustering is able to partition similar time episodes of driving into same groups, each describing a distinct driving scenario. We also tried to investigate whether it is possible to deduce meaningful scenarios interpretations based on the clustered episodes and verified them using camera
images that we mapped to each driving scenario. In order to evaluate the similarity between extracted episodes we measured the Pearson correlation and their probability distribution. We conducted an empirical evaluation using vehicle bus signals of mainly four vehicle sensors measuring the acceleration pedal, braking pressure, steering angle and the vehicle speed. For future work, we intend to improve this approach by considering object detection using artificial neural networks to provide additional inputs to the clustering and be able to come up with more detailed interpretations. We also plan to try other clustering algorithms and to compare the obtained outcome.

References


RoBF: An Auto-Tuning Bloom Filter for Mixed Queries on LSM-tree

Ruicheng Liu, Peiquan Jin, Shouhong Wan, Bei Hua

1School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
2Key Lab. of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei, China

jpq@ustc.edu.com

Abstract—Bloom filter is an efficient technique to improve query performance in LSM-tree-based databases, such as RocksDB, HBase, and Cassandra. However, the original Bloom filter uses a fixed false positive rate (FPR), which makes it inefficient for mixed queries that involve both point and range queries. To solve this problem, in this paper, we present an improved Bloom filter called RoBF (Range-Query-Oriented Bloom Filter), which uses a mixture of Bloom filters and can process mixed queries on LSM-tree efficiently. We design an efficient algorithm for generating the solution based on the query distribution. We compare our proposal with the trie-based filter and find out that each has its own advantages for various scenarios. Therefore, we propose to use different filters with varied sizes for different levels on LSM-tree. Following this idea, we present an algorithm to generate specific filters with a specific size for different levels on LSM-tree to optimize the performance of mixed queries under limited memory space. We conduct comparative experiments and compare the proposed RoBF with various competitors, and the results show that RoBF can improve the performance of evaluating mixed queries by up to 6x to 30x, compared to the original Bloom filter in RocksDB.

Keywords—Mixed query, Bloom filter, LSM-tree

I. INTRODUCTION

LSM-tree [1] has been widely used in many key-value data stores due to its high write performance. There are many database based on LSM-tree, for example, Cassandra [2], HBase [3], and Rocksdb [4]. One of the main challenges for such databases is to avoid query performance degradation caused by the multi-level write buffers of LSM-tree [5].

To accelerate the query performance of LSM-tree, many researchers proposed to add Bloom filters into the LSM-tree structure. Bloom filter is an effective scheme with only false positive errors and no false negative errors, which is ideal for reducing the query amplification in the LSM-tree [6]. Presently, Bloom filters have been used in many index structures to accelerate query performance, such as BloomTree [7].

However, although Bloom filters can improve query performance for point queries that aim to retrieve individual key-value pairs, they are not efficient for evaluating range queries. So far, researchers have proposed the Prefix Bloom filters [8] to record a fixed-length prefix for each key to respond to range queries. For example, if we want to know if an SST (Sorted String Table) contains a key between [Hello, Henry], we can simply return false if we are certain that the SST does not contain a string that begins with "He". However, the PBF scheme will lower the accuracy of point queries.

In this paper, we study the limitations of Bloom filters in handling mixed queries and propose a new solution. Briefly, the contributions of this study are four-fold.

(1) First, we found that using both of these filters can produce better results for mixed queries, and we call this filtering scheme DBF (Double Bloom Filter). Thus, we propose an algorithm to dynamically determine the parameters of the DBF, including the length of the prefix and the memory usage ratio of the two filters, which achieve ideal results on a particular query distribution.

(2) Second, to achieve high performance on a more general query distribution, we propose RoBF (Range-Query-Oriented Bloom Filter), which extends the number of prefix bloom filters from one to many. We extend the previous algorithm to be used to determine the parameters of RoBF. Similar to DBF, RoBF supports both point and range queries, and in most scenarios comprehensive queries are nearly twice as good as DBF. In comparison with SuRF, a trie-based hybrid query filter, RoBF has a significant performance advantage in small memory.

(3) Third, we found that for the same LSM-tree and the same query distribution, different levels of SST actually have completely different data and query distribution, meaning that different filters with different sizes should be applied to the data of different levels. Based on this idea, we propose an algorithm to determine the parameters for each level, including filter types and memory consumption.

(4) Finally, we conduct comparative experiments on a 100GB dataset and compare the proposed RoBF with various competitors, and the results show that RoBF can improve the performance of evaluating mixed queries by up to 6x to 30x, compared with the original Bloom filter of RocksDB.

II. RELATED WORK

There are several optimizations for range queries for Bloom filters, such as the Prefix Bloom Filter (PBF) [8]. Compared with Bloom Filter, PBF uses a fixed-length key prefix to query keys, reducing the performance of point queries in exchange for support for range query filtering. PBF has long existed in RocksDB as an experimental function, and has officially become one of the main functions of RocksDB in recent years.
Unlike Bloom filters, PBF hashes each prefix of key length \( L \) in the set, rather than the entire key. PBF queries the prefix of target key length \( L \) in the point query, which reduces the accuracy of the point query to some extent. However, PBF fails to support prefix queries with the prefix length greater than \( L \).

We compare the performance of RoBF with that of PBF in our experiments, and find that in most cases, the filtering performance of PBF is at least half of that of RoBF, and in many cases even close to that of RoBF. Because PBF has fewer parameters and is easy for database administrators to adjust, we think it is a useful filter.

SuRF is a recently proposed filter that employs fast succinct tries [9]. SuRF is a trie-based filter whose basic idea is to store the exponentially expanded part of trie and other parts separately to achieve higher compression rate. SuRF has the highest performance of trie-based filters that I know of. We regard SuRF as the representative of trie-based filters. Therefore, we often compare RoBF and SuRF in experiments.

Trie-based filters generally save prefix information with length not less than \( y \), that is, for every key in \( S \), the prefix with key length \( L \) will be fully recorded in the trie-based filter. In this way, the filter will always correctly answer interval queries \([a,b]\) if neither \( a \) nor \( b \) is longer than \( L \); Even if the length of \( a \) or \( b \) exceeds \( L \), the filter can be replaced with the result of \([a',b']\), where \( a' \) and \( b' \) are prefixes for the length of \( a \) and \( b \) not exceeding \( L \), respectively.

The disadvantages of this filter are twofold. On the one hand, if the filter wants to answer a prefix query with length \( L \), it needs to keep the \( L \) bits before each key completely, which sometimes consumes unnecessary space. On the other hand, trie-based filter performance is very sensitive to space, and its accuracy will be low if the space does not reach a certain threshold. In general, trie-based filters perform well when space exceeds a threshold, but not all scenarios tolerate such a high memory consumption filter.

Monkey [10] is another Optimal navigable key-value store. Monkey discusses the need to apply memory filters of different sizes to different levels. According to the characteristics of Bloom Filter, Monkey provides the optimal solution for point-only query scenarios. We extend this idea to the mixed query scenario and select different filters for different levels to optimize the mixed query performance.

III. DESIGN OF ROBF

A. Motivation

We have mentioned the design of two filters. The first one is Hash-based filters, such as PBF [8], which record the hash information of prefixes to support prefix queries with length no less than \( L \). This kind of filters is designed for small memory scenarios. Another one is trie-based filters, such as SuRF [9], which record the complete information of prefixes to support range queries for large memory scenarios.

We show the differences between the two filters in Fig. 1(a), which shows the distribution of mixed queries consisting of three types of prefix queries and point queries. Each horizontal partition represents a prefix query or a point query, and the horizontal width represents its percentage in the query. In a partition, the top half represents the hit rate of the query, and the bottom half represents the miss rate, which means that a good filter always covers as much of the lower part of the figure as possible. Figure 1(f) shows a trie-based filter that can accurately answers queries with a prefix length less than \( L \), but it has an error rate for queries with a prefix length greater than \( L \). Figure 1(b) shows a hash-based filter that can answer a prefix query with length greater than \( L \), but it cannot answer a prefix query with length less than \( L \).

Note that the range query filter may have three types of errors:

1. Trie-based filter will generate false positives when answering queries with length greater than \( L \), but the frequency of such false positives will decrease with the increase of memory;
2. Hash-based filters generate false positives due to hash conflicts. The frequency of such false positives also decreases as memory increases;
3. Hash-based filters generate false positives when replacing interval queries with prefix queries. Such false positives are inherent and cannot be corrected by adding more memory. This type of errors cause hash-based filters to encounter performance bottlenecks when memory is large enough.

B. Data Structure

Before introducing RoBF, we first explain the working process of hash-based filters.

Figure 1(c) shows how PBF works. It can only respond to point queries and prefix queries with length \( L \) or greater. Figure 1(d) shows how a DBF works, which is essentially a hybrid filter consisting of a PBF and a BF. Range queries need to be tested by PBF, while point queries need to pass both PBF and BF tests. RoBF is an improved version of DBF that contains multiple PBFS and one BF. Figure 1(e) shows how RoBF works. RoBF can be regarded as a number of prefix hash filters with different lengths, each of which has a prefix length ranging from 1 to a threshold \( L_{\text{max}} \). Here, \( L_{\text{max}} \) represents the upper bound of the key length. Similar to Bloom filters, each test uses a different hash function, and all test results are stored in the same hash table.

In general, BF is more suitable for scenes with a large proportion of point queries, and PBF is more suitable for scenes with a fixed width of range queries. In addition, DBF can be dynamically adjusted according to the proportion of point queries and range queries. RoBF needs to decide the filter parameters according to the query distribution information. Generally, BF, PBF, and DBF can be considered as a special case of RoBF. In practical applications, we need to choose a right filter according to the load characteristics and determine the parameters of the filter.

C. Algorithm of Generating RoBF

In this section, we present the algorithm of generating RpBF, which is named RoBF-generator. Basically, the opti-
Fig. 1. **Performance of different filters** - Each column represents one type of query. The width of each column represents the weight of the query. The height of the gray rectangle represents the percentage of queries that are hit, and the height of the white rectangle represents the percentage of queries that are missed. The shaded portion represents the filtered query. Each shaded section, except for Figure 1(f), represents a hash test.

**Fig. 2. Four kinds of hash-based filters with seven hash tests.** Each grid represents a test, the number represents the prefix length of the test, and $L$ represents the length of the key. For example, DBF contains four hash tests for prefixes with length 13 and three hash tests for full key.

<table>
<thead>
<tr>
<th>Filter Name</th>
<th>Hash Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bloom Filter</td>
<td>L L L L L L L</td>
</tr>
<tr>
<td>Prefix Bloom Filter</td>
<td>13 13 13 13 13 13 13</td>
</tr>
<tr>
<td>Double Bloom Filter</td>
<td>13 13 13 13 L L L</td>
</tr>
<tr>
<td>RoBF</td>
<td>11 13 13 14 L L L</td>
</tr>
</tbody>
</table>

The optimization of filters means that there are as few false positives as possible, that is, the lower half of Fig. 1(e) should be covered as much as possible. For this purpose, we need to not only understand the distribution of the query, but also collect the characteristics of the dataset. For the sake of efficiency, we always assume that for any queries with prefixes whose length is $p$, the total number of the prefixes with length $p$ is always close to the total number of the keys. Such an assumption allows us to make a general filter parameter recommendation without knowing the key distribution for each different SST.

We first use a simplified version of RoBF-generator to set the parameters of the DBF. When we try to set length $p$ as the prefix length for DBF, we need to determine how much memory PBF and BF use. Here, the theoretical FPR is a convex function relative to the memory usage of a filter, and this conclusion can be used to speed up the search speed of the algorithm.

Furthermore, in the LSM-tree, each write buffer level actually has a separate distribution of query requests, even though the global query is the same for each level. For example, in the LSM-tree, it is likely that all keys in an SST of the underlying buffer share the same 10-byte prefix, which means that prefix queries with length less than 10 will be filtered out during range checking. However, this case will not happen in the upper buffer.

**D. BPK-Balancer Algorithm**

We present a new algorithm called BPK-balancer to search filter parameter combinations for each write buffer level for finding the best parameter balance. The term $BPK$ is the abbreviation of BitsPerKey. At the beginning of the algorithm, we provide an optimal solution to accelerate the search speed, and this solution is based on the conclusion in Monkey [10].
Algorithm 1: RoBFGenerator (Current, Tests, Filtered)

```
input : Queries
output: BestSolution
1 Function Search (Current, Tests, Filtered):
2     if Current = Length then
3         X = Solution.Set(Queries, Current, Tests);
4         if Solution > BestSolution then
5             BestSolution = Solution;
6     end
7     return X;
8 end
9 MaxCovered = 1;
10 TernarySearcher.Set(0, Tests);
11 for i ∈ TernarySearcher do
12     X = Solution.Set(Current, i);
13     Covered = X * Missed.Sum(Current, Length);
14     Rest = Search(Current+1, Test-i, Filtered+X);
15     Covered = Covered + Rest;
16     TernarySearcher.AddSolve(i, Covered);
17     if Covered > MaxCovered then
18         MaxCovered = Covered;
19     end
20 end
21 return MaxCovered;
```

Algorithm 2: BPK-Balancer (Current, Bits)

```
input : BitsPerKey
output: BestSolution
1 Function WaitForTest (Level, Bits):
2     if FilterRecord.Lookup(Level, Bits) = False OR FilterRecord.Time(Level, Bits) > TimeLimit then
3         for type ∈ FilterSet do
4             SetFilter(Level, type, Bits);
5             FPR = RealWaitForTest(Level);
6             FilterRecord.Update(Level, Bits, FPR);
7         end
8     end
9     return FilterRecord.Latest(Level, Bits)
10 Function Balancer (Level, Bits):
11     if Current = 0 then
12         X = Solution.Set(Current, Bits);
13         if Solution > BestSolution then
14             BestSolution = Solution;
15         end
16         return X
17 end
18 MinFPR = 1;
19 TernarySearcher.Set(0, Bits);
20 for i ∈ TernarySearcher do
21     X = Solution.Set(Current, i);
22     FPR = WaitForTest(Current, i);
23     Rest = Balancer(Current - 1, Bits - i);
24     TernarySearcher.AddSolve(i, FPR + Rest);
25     if FPR + Rest < MinFPR then
26         MinFPR = FPR + Rest;
27     end
28 end
29 return MinFPR
```

IV. PERFORMANCE EVALUATION

A. Workload

The workload we used in the experiment is similar to the workload in YCSB [11], except that the queries in our workload include half point queries and half range queries. The maximum interval length of the range query is set to 100 by default, and the interval length distribution is uniform. The keys used in the database are strings with length 16, and the values of the keys are set to no more than 10^{14} positive integers. All the keys follow a uniform distribution. In addition, we set the size of the LSM-tree to contain 3^9 key-value pairs, with a total key size of about 50GB.

In the test of a single SST, we set the hit rate of point query and range query at about 10%. In the test of the LSM-tree, we set the global point query hit rate and range query hit rate at about 10%. When testing a single SST, we set the BPK (BitsPerKey) to 24, which is three times the default value, because it is convenient for us to compare the characteristics of RoBF and SuRF. For the global testing, we set the BPK to 8-16, which is 1-2 times the default value.

B. Filters Compared

We mainly compare our RoBF with the following filters in the experiment.

- **Prefix Bloom Filter (PBF)** [8]. We will not use the basic Bloom Filter as a comparison, because our test contains a large number of range queries, and we will use PBF instead.
- **Double Bloom Filter (DBF)**. We use only one PBF and one BF to explore whether it is necessary for RoBF to use multiple parameters.
- **Perfect Prefix Filter (PPF)**. PPF is a filter that can theoretically answer any prefix query accurately. We construct such a Filter to observe the high performance limit of the hash-based Filter.
- **SuRF-Real (SuRF)** [9]. SurF is a trie-based filter based on FST (Fast Succinct Trie). For most SST, SuRF cannot be created at the BPK value less than 15.

C. Results

Figure 3 shows the relationship between the false positive rate of different filters and BPK.
As we can see, only SuRF finally surpasses the Perfect Prefix Filter (PPF) with the increase of memory. That is to say, in this test, when the memory size of the filter exceeds 70 BPKs, that is, the average memory allocated per key is greater than 8.4 bytes of space, the performance of the hash-based filter will hardly surpass that of the trie-based filter.

This is due to the third type of false positives mentioned earlier. Since hash-based filters can only process prefix queries, there is an inherent false positives rate. With the increase of BPK, the performance of RoBF gradually converges to PPF, and the false positives rate cannot be further reduced, while trie-based does not have such a problem.

After the BPK reached 70, the false positive rate of SuRF decreased significantly, which is also the characteristic of trie-based filter. Such a filter must increase the length of the stored prefix in order to effectively support a small range of queries, rather than simply adjusting the position of the test points as with a hash-based filter, which often means higher memory consumption.

In practice, most filters cannot provide more than 8 bytes per key. It can be seen from the first half of the curve that the false positive rate of PBF/DBF/RoBF decreases obviously when BPK increases from 8 to 20, which reflects the feature that the hash-based filter is suitable for small memory. In contrast, DBF has relatively close performance to PBF and relatively poor performance relative to RoBF, which means that RoBF has more parameters than DBF that are not redundant.

As memory increases further, the false positive rate of PBF and DBF no longer changes significantly, but the false positive rate of RoBF can still decrease further, because RoBF can use memory to support a range of queries of different lengths, rather than only reducing the false positive rate of individual prefix queries.

As shown in Fig. 1, We can see the advantages and disadvantages of the two types of filters: the trie-based filter must continuously store the prefix information of the key, otherwise it cannot be guaranteed that there is no false negative error, so the trie-based filter always needs some space to store the prefix information that is of no value for the reply; On the other hand, the hash-based filter, even if it can answer the prefix query perfectly, does not improve the hit ratio of the range query further, thus causing performance bottlenecks when memory is large enough.

Figure 4 shows the results of our second experiment. We compared the false positive rates of different filters at different levels of LSM-tree. In order to show the characteristics of the filter more intuitively, we randomly selected a SST from each level and showed its FPR.

As can be seen from the figure, the false positives rate of SuRF decreases first and then rises with the increase of Level, which means that the trie-based filter has poor performance in both the upper and lower levels, but the reasons are different: in the upper level, the distribution of keys is relatively sparse, which means that SuRF needs more space to store prefix information that is not helpful for query; In the lower level, the range query requires the last few characters to respond correctly, and the trie-based filter requires a higher BPK to better support such a small range query.

The performance of RoBF is relatively stable, but there is a significant drop in performance in the last two levels. This is due to the fact that as the keys become denser, there are fewer interval queries that can be filtered by the prefix, i.e. the proportion of type 3 errors increases.

As mentioned earlier, allocating more memory for each filter in the upper write buffer can improve query performance. This is due to the fact that there is less data in the upper write buffer and more queries, and the same amount of memory can reduce more false positives for queries.

Taking the case of BPK=12 as an example, a basic scenario is to assign a filter of 12 BPKs to each file, in which case the average false positive rate per file is 0.6%; But if we map the keys in levels 0-2 directly to memory, and use SuRF filters with BPK of 78 and 34 in levels 3-4, respectively, and RoBF filters with BPK of 9 in level 5, the average memory...
Fig. 5. The optimal filter configuration scheme under different BPK - The figure shows optimal solutions for five scenarios. For example, when BPK is 12, Level-3 is recommended to use SuRF with 78 BPKs. "MAP" represents a full mapping strategy, while 128 represents a total key length of 16 bytes.

collection per file still does not exceed 12 BPKs, and Fig. 5 shows that the false positives rate drops to 0.11%, a reduction of 79.74%.

This conclusion is consistent with the law of practice. In the case of RocksDB, when RocksDB is configured, the index of the upper SST tends to have higher access frequency, which means that the index of the upper SST is actually resident in memory as long as we can provide sufficient cache space.

Since the upper-level SST filters use more memory, the memory usage of the lower-level SST is significantly lower than average, and the performance benefits of RoBF in small-memory scenarios are easier to realize. It is worth noting that SuRF is used as a middle level filter, which is consistent with our analysis in the previous section, i.e., trie-based filters perform poorly when the key distribution is too sparse or too dense.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a multi-parameter filter RoBF to improve the range queries of Bloom filters in LSM-tree. We propose an algorithm to determine the parameters of RoBF based on query distribution. We conducted comparative experiments to compare RoBF with various filters, and the results suggested the efficiency of RoBF. RoBF adopts multi-filter strategy, because we believe that different filters should be used in different levels of LSM-tree. Particularly, the lower the level, the higher the BPK provided for the filter should be. We designed an algorithm to search for the optimal solution. The results of this algorithm confirm our conjecture that the optimal solution generated by the algorithm can reduce the global FPR to 20% of the static solution.

At present, we determine the type and parameters of the filter through the data distribution and the memory size of each level in LSM-tree, while the memory size of filter in each level is determined by another algorithm. Two independent algorithms will lead to the calculation of the optimal solution takes a long time, which may affect the real-time performance of the parameters of the filter. In the future work, we plan to integrate the two algorithms together and adjust the parameters with a faster algorithm, which can also provide better theoretical support for the experimental results of this work. Also, we will extend RoBF to make it suitable for persistent

memory-based key-value stores [12], e.g., integrating RoBF with the persistent memory-friendly adaptive Radix tree [13] or B+-trees [14].

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation of China (No. 62072419). We also thank the anonymous reviewers for for their suggestions and comments to improve the quality of the paper. Peiquan Jin is the corresponding author.

REFERENCES

Abstract—Formula retrieval is an important research topic in Mathematical Information Retrieval (MIR). Most studies have focused on comparing formulae to determine the similarity between mathematical documents. However, two similar formulae may appear in completely different knowledge domains and have different meanings. Based on N-ary Tree-based Formula Embedding Model (NTFEM), we introduce a new hybrid retrieval model combining formula with its surrounding text for more accurate retrieval. Using keywords extraction technology, we extract keywords from text around the formula which can supplement the semantic information of formula. Then we get the representation vectors of keywords by FastText N-gram embedding model, and the representation vectors of formulae by NTFEM. Finally, documents are first sorted according to the similarity of keywords, and then the ranking results are reordered according to the formulae similarity. Experimental results show that the accuracy of top-10 results is at least 20% higher than that of NTFEM and can be 50% in some specific topics.

Index Terms—Mathematical Information Retrieval, Formula Similarity, Formula Embedding, Word Embedding, Keywords Extraction.

I. INTRODUCTION

Nowadays, the retrieval methods for linear sequence text are widely developed and used, such as Google, Baidu, Bing and other search engines [1]. However, these methods do not work well for complex formulæ that frequently appear in mathematical documents. In this case, formulæ-based embedding models are proposed for solving the retrieval problem of formulæ with two-dimensional structures. The models can capture the structural features of mathematical formulæ, but may lack semantic interpretation. Namely, formulæ with similar structure may appear in completely different knowledge domains and have different meanings. Therefore, the retrieval results may be unsatisfactory. It is a difficult problem that need to be tracked.

Generally, the text around the formula is typically a very good indication of what domain application the formula is used for. In text-based retrieval methods, documents are represented by a group of keywords, as keywords can summarize the information of text [2]. Then the similarity between two documents is calculated by keyword matching algorithms. More specifically, using keywords instead of long text as the retrieval units, can not only reduce the storage, but also improve the retrieval accuracy. Therefore, the additional semantic information of the formula can be supplemented through the keywords of the text around the formula.

In this paper, we present a hybrid retrieval model combining the formula with its surrounding text for more accurate Mathematical Information Retrieval (MIR). We first use Rapid Automatic Keyword Extraction (RAKE) algorithm [3] to extract the keywords from text around the formula, and the representation vectors of the keywords are then obtained by FastText N-gram embedding model [4]. Meanwhile, we get the representation vectors of formulæ by NTFEM [5]. Finally, the mathematical documents are first sorted based on the similarity of the keyword vectors, then the retrieval results are reordered according to the formulæ similarity. Experimental results on the dataset provided by TopicEq [2] show our model has achieved more accurate retrieval results than other retrieval models and can effectively capture semantic features of formulæ.

The remainder of this paper is organized as follows. We first introduce the preliminaries and challenges of formula retrieval in Section II. In Section III, we then present the hybrid retrieval model combining the formula with its surrounding text. In Section IV, we evaluate our model on the dataset provided by TopicEq and compare with other retrieval models. Section V reviews related work on formula retrieval methods and keyword extraction methods. Finally, we conclude this paper in Section VI.

II. FORMULA RETRIEVAL

A. Definition of Formula Retrieval

Most present information retrieval systems usually do not consider mathematical notations and formulæ in documents because they cannot build effective indexes for them. Search engines like Google mostly treat user inputs of symbols, equations and formulæ as normal text without understanding their mathematical semantics [6]. They may be able to find similar text, but are very often fail to find the exact match. Given a query formulæ, the system should be able to parse its structure and semantics and then find the matching documents with similar formulæ. For instance, given an incomplete equation \( e^{\pi n} + 1 \), the formula retrieval system
should match to the Euler’s Identity formula: $e^{i\pi} + 1 = 0$.

The key issue is how to measure the similarity between two formulae which is different from the text similarity. We need extract features of math formulae so that we can distinguish from different formulae, then get similar formulae.

### B. Challenges of Formula Retrieval

Formulae are generally displayed in two-dimensions. However, the current representation ways, such as \TeX and MathML, cannot reflect the structural characteristics of formulae. Besides, formulae are highly abstract. Two similar formulae may appear in completely different knowledge domains and have different meanings, and this lead to unexpected match [2]. Therefore, both structural and semantic features of formulae should be considered in retrieval process.

#### II. BLACK HOLES IN EINSTEIN GRAVITY

As a warming exercise, in this section, we will briefly review the observation made by Padmanabhan [1] by generalizing his discussion to a more general spherically symmetric case. In Einstein’s general relativity, the gravitational field equations are

$$ G_{\mu\nu} = R_{\mu\nu} - \frac{1}{2} R g_{\mu\nu} + 8 \pi G T_{\mu\nu}, $$

where $G_{\mu\nu}$ is Einstein tensor, $R_{\mu\nu}$ is the trace-free part of the Riemann tensor, $R$ is the scalar curvature of the metric, and $T_{\mu\nu}$ is the stress-energy tensor of matter field. On the other hand, for a general static, spherically symmetric spacetime, the metric can be written down as

$$ ds^2 = dt^2 - a^2(t) \left[ \frac{dr^2}{r^2} + r^2 d\Omega^2 \right], $$

where $a(t)$ is a function of time $t$.

An interesting observation is, with text around the formula, the meaning of the formula is determined more clearly. As shown in Fig. 1, the highlighted words are keywords extracted by keywords extraction algorithms, and the word relativity and term gravitational clearly show the formula is intended for Physics. This example shows that keywords can greatly help formula retrieval, and the combination of formula retrieval and context analysis can better capture the semantics of documents and lead to more accurate match.

#### III. METHOD

The process of our hybrid retrieval model is shown in Fig. 2. And the detail of process is explained as follows:

1) **Processing of mathematical retrieval model**: Formulae are parsed into n-ary trees, and keywords are extracted from text around the formula.

2) **The representation vectors of formulae and keywords**: The representation vectors of keywords are obtained by an n-gram word embedding model, while the representation vectors of the formulae by NTFEM.

3) **Processing of queries**: In the same way, the representation vectors of keywords and formulae from input queries are calculated respectively by above models.

4) **Optimization for retrieval results**: The mathematical documents in the database are sorted based on the similarity of the keywords vectors first, after that, the retrieval results are sorted again on the basis of formula similarity.

#### A. Formula Embedding Model

NTFEM first transforms the mathematical formula from a two-dimensional structure to a one-dimensional linear sequence. The transformation steps are listed as follows:

- Convert a formula (MathML) into an n-ary tree.
- Generate the tuple sequences and tokenize the tuples.

Fig. 3 shows the process of the formula $a + b \times c + 2 \times b \times c$ being converted from MathML notation to tree structures, in which $b$ is the binary tree, and $c$ is the n-ary tree.

![Fig. 3. Example of formula conversion](image)

Then NTFEM sets labels for symbols in the formulae, where these labels fall into the following categories:

- Numbers “N”.
- Identifiers such as variable symbols “V”.
- Commutative operators “U”.
- Non-commutative operators “O”.

As shown in the Table I, NTFEM uses a pair-based method [7] to define "words" and generate "sentences" by breadth-first traversal which is the input of the embedding model.

In order to better represent the features of formulae, NTFEM uses the weighting strategy based on FastText N-gram embedding model [4].
1) Level Weight: Taking depth and complexity into consideration, the weight of tuple $t$ can be expressed in the formula sequence $f$ with

$$W_{f[t]} = \frac{(\text{depth} - D(t)) \times \alpha + \sum_{i}^{\text{depth}} L(i)}{\text{depth}} \times \beta,$$

where $\text{depth}$ is the depth of the n-ary tree of $f$, $D(t)$ is the level of tuple $t$ in the n-ary formula tree, and $L(i)$ is the number of tuples at the $i$-th level, and $\alpha, \beta$ are two tuning parameters. Here we set $\alpha + \beta = 1$, and adjust these two parameters to get better results.

2) Frequency Weight: With consideration of tuple frequency in the corpus, NTFEM combines SIF [8] with level weight to get the final formula embedding $v_f$:

$$v_f = \frac{1}{|f|} \sum_{t \in f} \gamma + p_t \gamma w_t,$$

where $\gamma$ is hyperparameters, $p_t$ is the frequency of tuple $t$ in corpus, and $w_t$ is the formula tuple embedding.

### Extract and Train the Keywords

In this paper, we use RAKE algorithm [3] to extract the keywords from text around the formula. Contrary to other methods that rely on Natural Language Processing (NLP) technologies, RAKE can automatically extract keywords from text with only one traversal. Moreover, it can extract key phrases from mathematical text, especially longer technical terms, which is in line with the task scenario of this paper. The algorithm first uses punctuation marks to break a document into clauses, and then, for each clause, uses stop words as delimiters to divide the clauses into phrases that serve as candidates for the final extracted keywords. Next, each phrase can be divided into several words by space. And each phrase can be scored by the sum of its word scores. The score of word $w$ is calculated as follows:

$$\text{WordScore}(w) = \frac{\text{Degree}(w)}{\text{Frequency}(w)} \gamma,$$

where $\text{Degree}(w)$ is the degree of the word (a concept in the network) and $\text{Frequency}(w)$ is the frequency of the word. Finally, the top third of candidate phrases are identified as keywords, after the phrases are sorted by their scores in descending order.

Like processing in NTFEM, FastText is also used for keywords training in this paper. In order to better capture the semantic features of keywords, we make the following improvements to the training process:

1) Stop words adjustment: Stop words such as by, allows, almost and everywhere appear frequently in documents and have little effect on reflecting useful information. In most models, these words would be removed in order to improve training efficiency. But for mathematical text corpus, stop words may appear in the definition of mathematical formula, the description of the mathematical theory, and other text which is important for reflecting mathematical semantics. Therefore, we have adjusted the stop words and preserved some of them that may affect the mathematical semantics.

2) Negative Sampling: In the CBOW model, a word $w$ is predicted by its context. Namely, for a given $\text{Context}(w)$, the word $w$ is a positive sample and the others are negative samples. Generally, 5 negative sample words will be selected for each $\text{Context}(w)$. The probability that the sample $w_i$ is selected is:

$$P(w_i) = \frac{f(w_i)}{\sum_{j=0}^{n} f(w_j)}.$$ (4)

Then positive and negative samples are represented with $c, s$ respectively. In this case, the results of the output layer can be normalized between $[0, 1]$. Compared with a group of negative samples obtained by random sampling, the objective function of the model is listed as follows:

$$F = \sum_{n=1}^{N} \left[ \log \left( 1 + e^{-\gamma(s, c_j)} \right) + \sum_{m \in M_{c_j}} \log \left( 1 + e^{\gamma(m, c_j)} \right) \right],$$ (5)

where $M_c$ represents a group of negative samples obtained by negative sampling. And $\gamma(S, C)$ is the evaluation function related to the word $S$ and its context $C$, it can be calculated as follows:

$$\gamma(S, C) = \frac{1}{|C|} \sum_{s' \in C} u_{s'}^T v_s.$$ (6)

Here $v_s$ represents the word vector of the word $S$, and $u_{s'}$ is the word vector of the word $s'$ in the $\text{Context}(S)$.

### Similarity of Mathematical Documents

After obtaining the representation vectors of the formulae and the surrounding text, we use cosine similarity of vectors as the basic ranking. The documents is first sorted according to the similarity between the keywords in the database and the user input. For the query document $q$ and a document $p$ in the database, with their text vector $V_q$ and $V_p$ respectively, the similarity is measured as follows:

$$\text{Sim}(p, q) = \frac{\sum_{i=1}^{n} (V_q)_i \times (V_p)_i}{\sqrt{\sum_{i=1}^{n} (V_q)_i^2 \times \sum_{i=1}^{n} (V_p)_i^2}}.$$ (7)

Then, top-k results are reordered according to the cosine similarity of the formulae. The final results are the similar candidate documents of the user input.
IV. EXPERIMENTS AND RESULTS

A. Dataset

For evaluation of our hybrid model, we use the dataset provided by TopicEq [2], which includes nearly 100,000 scientific and technological articles published in arXiv.org in the past five years and generate 400,000 pieces of data. Each data consists of a formula in \( \LaTeX \) notation and ten sentences around the formula, where five before the formula and the rest after it. The dataset covers 10 topics and labels them with \( T_1, T_2, \ldots, T_{10} \), including astrophysics (\( T_1 \)), relativity (\( T_2 \)), graph theory (\( T_3 \)), linear algebra (\( T_4 \)), machine learning (\( T_5 \)), quantum physics (\( T_6 \)), particle physics (\( T_7 \)), number theory (\( T_8 \)), optimization (\( T_9 \)) and probability (\( T_{10} \)). We randomly select 93,051 pieces of data from 5 categories and used 90% of the data as the training set and the rest as the test set. The distribution of the data is listed as follows:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Training Set</th>
<th>Test Set</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>astrophysics</td>
<td>18,401</td>
<td>1,810</td>
<td>20,217</td>
</tr>
<tr>
<td>relativity</td>
<td>18,826</td>
<td>1,823</td>
<td>20,649</td>
</tr>
<tr>
<td>graph theory</td>
<td>17,469</td>
<td>1,678</td>
<td>19,147</td>
</tr>
<tr>
<td>linear algebra</td>
<td>16,522</td>
<td>1,652</td>
<td>18,174</td>
</tr>
<tr>
<td>machine learning</td>
<td>13,048</td>
<td>1,240</td>
<td>14,828</td>
</tr>
</tbody>
</table>

Note that we have made a tool to convert the formula with notation from \( \LaTeX \) to MathML. In this case, we can obtain the representation vectors of formulae through NTFEM.

B. Retrieval Results and Evaluation Standards

In this paper, we used \( P@k \) to calculate the accuracy of retrieval results. For a query, \( P@k \) represents the proportion of results related to user input among the top-k retrieval results, and the calculation formula is:

\[
P@k = \frac{\text{true positives} @k}{\text{true positives} @k + \text{false positives} @k}.
\]

Experimental results show that the retrieval accuracies after enhanced through the incorporation of surrounding text are greatly improved, compared with pure formula retrieval methods. As shown in Table III, for the topic of astrophysics, the accuracy of top-10 retrieval results of NTFEM-K is improved by 50% compared with NTFEM. Furthermore, retrieval accuracy in the topic of machine learning is much lower than others of NTFEM retrieval model. This may be because the field of machine learning intersects with other fields, and NTFEM has difficulty in distinguishing formulae with similar structures, which can be solved through the incorporation of the surrounding text.

<table>
<thead>
<tr>
<th>Topic</th>
<th>P@10</th>
<th>P@50</th>
<th>P@100</th>
<th>P@200</th>
<th>P@500</th>
</tr>
</thead>
<tbody>
<tr>
<td>astrophysics</td>
<td>0.400</td>
<td>0.320</td>
<td>0.300</td>
<td>0.315</td>
<td>0.324</td>
</tr>
<tr>
<td>relativity</td>
<td>0.300</td>
<td>0.300</td>
<td>0.260</td>
<td>0.285</td>
<td>0.286</td>
</tr>
<tr>
<td>graph theory</td>
<td>0.300</td>
<td>0.300</td>
<td>0.310</td>
<td>0.260</td>
<td>0.262</td>
</tr>
<tr>
<td>linear algebra</td>
<td>0.300</td>
<td>0.220</td>
<td>0.250</td>
<td>0.180</td>
<td>0.188</td>
</tr>
<tr>
<td>machine learning</td>
<td>0.700</td>
<td>0.500</td>
<td>0.430</td>
<td>0.310</td>
<td>0.250</td>
</tr>
</tbody>
</table>

Fig. 4. Precision and MaxF indicators of the above three models

TABLE III

<table>
<thead>
<tr>
<th>Topic</th>
<th>NTFEM</th>
<th>NTFEM-K</th>
<th>NTFEM</th>
<th>NTFEM-K</th>
<th>NTFEM</th>
<th>NTFEM-K</th>
<th>NTFEM</th>
<th>NTFEM-K</th>
<th>NTFEM</th>
<th>NTFEM-K</th>
</tr>
</thead>
<tbody>
<tr>
<td>astrophysics</td>
<td>0.300</td>
<td>0.800</td>
<td>0.260</td>
<td>0.780</td>
<td>0.280</td>
<td>0.770</td>
<td>0.325</td>
<td>0.630</td>
<td>0.310</td>
<td>0.454</td>
</tr>
<tr>
<td>relativity</td>
<td>0.300</td>
<td>0.500</td>
<td>0.280</td>
<td>0.400</td>
<td>0.230</td>
<td>0.430</td>
<td>0.255</td>
<td>0.415</td>
<td>0.278</td>
<td>0.354</td>
</tr>
<tr>
<td>graph theory</td>
<td>0.300</td>
<td>0.500</td>
<td>0.240</td>
<td>0.480</td>
<td>0.250</td>
<td>0.400</td>
<td>0.225</td>
<td>0.345</td>
<td>0.228</td>
<td>0.316</td>
</tr>
<tr>
<td>linear algebra</td>
<td>0.200</td>
<td>0.900</td>
<td>0.220</td>
<td>0.740</td>
<td>0.170</td>
<td>0.680</td>
<td>0.165</td>
<td>0.675</td>
<td>0.158</td>
<td>0.598</td>
</tr>
<tr>
<td>machine learning</td>
<td>0.100</td>
<td>0.900</td>
<td>0.180</td>
<td>0.560</td>
<td>0.150</td>
<td>0.510</td>
<td>0.215</td>
<td>0.420</td>
<td>0.202</td>
<td>0.294</td>
</tr>
</tbody>
</table>

TABLE IV

<table>
<thead>
<tr>
<th>Topic</th>
<th>P@10</th>
<th>P@50</th>
<th>P@100</th>
<th>P@200</th>
<th>P@500</th>
</tr>
</thead>
<tbody>
<tr>
<td>astrophysics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>relativity</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>graph theory</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>linear algebra</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>machine learning</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In this paper, we also study the effect of embedding all surrounding text without extracting keywords, which is called NTFEM-T model. We find that the overall performance of NTFEM-T is also better than NTFEM on retrieval accuracy. As shown in Table IV, the retrieval accuracies of NTFEM-K and NTFEM-T are much higher than that of NTFEM for the machine learning topic. This means that the text around the formula can supplement the semantics of the formula. In addition, for retrieval tasks on topics relativity and graph theory, the retrieval accuracies of NTFEM-T are about 20% lower than those of NTFEM-K, which shows that the keywords extraction method can efficiently capture the semantic features of the formula.

We use two indicators Precision and MaxF to compare the performance of the above models. As shown in Fig. 4, the precision of NTFEM-K is much higher than other models, especially in retrieval tasks of $T_1$, $T_3$ and $T_4$ topics. MaxF is the harmonic value of precision and recall, which can reflect the overall performance of the model. It is easily observed that the NTFEM-K model has best performance.

Finally, we evaluate the above models on the task of document clustering. A heat map of confusion matrix (Fig. 5) is used to visualize the clustering results. The diagonal elements of the heat map indicate the number of documents which are correctly labelled. According to the experiment results, NTFEM-K has also achieved better performance on the classification task compared to the NTFEM model.

V. RELATED WORK

The research on Math Information Retrieval has been a hot topic in the fields of information retrieval and knowledge engineering. In the early stage, formula-based methods used various transformation algorithms to convert the formulae into special notations, which are suitable for indexing and retrieved by matching algorithms. Later, the text around the formula were incorporated to supplement the information of the formula.

A. Formula-based Retrieval

For pure formula retrieval, most previous studies can be roughly categorized into text-based and tree-based models [9], [10]. In text-based methods, formulae are converted to ordered strings, which can be considered as inputs with the same processing in the traditional text retrieval models. Math Indexer and Searcher (MIAS) [11] implemented formulae matching in three steps: ordering commutative operations, variables unification and constants unification. DLMF project [12] implemented the textualization of math formulae through flattening and normalization process, in which each formula generates a unique form for all possible orders of operator symbols. Kumar et al. [13] applied the largest common sub-string algorithm to calculate the similarity between documents. Unfortunately, most text-based methods are inefficient, and the retrieval accuracy is low due to the inability to extract the two-dimensional structural features of the formula.

In tree-based methods, the formulae are transformed into trees and the partial matching methods are usually used for retrieval tasks. For example, MathWebSearch (MWS) [14], a model based on the substitution tree index, used term indexing to minimize access time and storage. Moreover, different representation trees of formulae also affect retrieval performance. Zanibbi et al. compared the performance based on two hierarchical representations, Symbol Layout Trees (SLTs) and Operator Trees (OPTs), and designed a series of mathematical retrieval systems [15].

In recent years, some embedded models have been applied to the field of mathematical information retrieval. Thanda et al. [16] first used the Doc2Vec model on mathematical formulae by representing the formula tree as a 100-dimensional
vector. Gao et al. [1] proposed symbol2vec and formula2vec, where symbol2vec simply learns the symbols of formulae in \( \LaTeX \) based on CBOW by using negative sampling, and formula2vec treats the formulae as sentences and uses the PV-DM to learn the characteristics of formulae.

B. Combination of Formula and Its Surrounding Text
Compared to the pure formula retrieval, we believe that the retrieval model combining formula with text can learn more useful features by extracting the mathematical semantics of the text around the formula. Krstovski and Blei [6] proposed a word embedding model for mathematical expressions. They treated the entire equation as a word through the distributed representation of the formula, and embedded it in conjunction with the surrounding text. However, they ignored the internal structure and symbolic semantics of formula. Yasunaga and Laferty [2] designed a topic model for scientific documents containing formulae. In this model, they used long and short-term memory models (LSTM) [17] to learn the characteristics of the formula sequence. Through a series of verifications, the effect of the topic model that combines formula and the surrounding text is better than that of a simple text topic model. Most previous hybrid models handle formulae in a rough way, so that they cannot capture the characteristics of formulae well. Moreover, the research on the combination of formula and the surrounding text is still in its early stage and cannot be effectively applied to specific retrieval tasks.

C. Keywords Extraction
Keywords use a set of words to define the content of the text, which can reduce storage space and streamline the calculation of document similarity. Generally speaking, there are two ways to generate keywords:

- Keyword allocation technique selects multiple words in a given thesaurus as keywords to label a document.
- Keyword extraction technique extracts some words from a document as keywords to label it.

At present, the more prevalent method is keyword extraction technique, which can be mainly divided into two categories:

1) Supervised learning algorithm: A supervised learning algorithm transforms the process of keyword selection into a binary classification problem. The specific method is to set a boolean label on the extracted candidate words to indicate whether the candidate word is a keyword, and then train the keyword classifier with a certain amount of datasets with labels. For a document, all candidate words can be extracted first, and then the trained keyword classifier is used to label and classify, and finally all the keywords are obtained.

2) Unsupervised learning algorithm: An unsupervised learning algorithm first extracts candidate words from the document, and scores the candidate words according to certain rules. Different scoring strategies focus on the features of different dimensions of a text, and then output top-\( k \) candidate words with the highest scores as the final keywords. The common algorithms are TF-IDF, RAKE, LDA, and etc.

VI. CONCLUSION
In this paper, we present a hybrid model NTFEM-K in corporation of surrounding text for mathematical information retrieval. We use NTFEM to produce formula embeddings and obtain the representation vectors of keywords by FastText that captures mathematical semantics. Then we use quadratic sorting method to obtain the retrieval results. Experiments show that our model achieves higher retrieval accuracy than previous models of formula retrieval.

ACKNOWLEDGEMENT
This work was partially supported by OneSmart Education Group and the Open Research Fund of Engineering Research Center of Software/Hardware Codesign Technology and Application, Ministry of Education (East China Normal University).

REFERENCES
Automatic Comprehension of Geometry Problems using AMR Parser

Anca-Elena Iordan
Department of Computer Science
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
anca.iordan@cs.utcluj.ro

Abstract—Automatic comprehension of geometry problems described in natural language is a crucial and challenging stage of numerous automatic geometry problem solvers. These systems should comprehend the existing information in natural language geometry problems with the purpose to extract the geometric relationships between elements and to accomplish automatic solutions using intelligent methods. Abstract Meaning Representation is a popular framework for annotating whole sentence meaning. This paper proposes the addition of a new feature to an existing transition-based AMR parser that constructs AMR graphs from statement of geometry problems described in English language. The new feature consists in explicit embedding of the coreference detection into the parser. Integrating the automatic comprehension method with different geometry systems will greatly enhance the efficiency and intelligence in automatic solving. This approach shows improvements over the best previously published systems for extract information from statement of geometry problems described in English language.

Keywords - Geometry Problems; Python; Natural Language Processing; AMR Parser

I. INTRODUCTION

The Abstract Meaning Representation (abbreviated AMR) [1] is a semantic representation of natural language text, which attempts to catch the connotation of a sentence into a structured AMR graph. The AMR domain retains who accomplishes what to whom in a sentence, and because to this argument, the collected information is kept in a rooted, directed, acyclic graph, appointed the AMR graph [2]. AMR graphs have tags on edges, which are acknowledged as relations, and the leaves of an AMR graph are watched as notions. AMR cannot depict coreferences which cross sentence limits and, likewise, it cannot discern amongst hypothetical or real events. The available mention-pair systems whichever identify coreferences [3] do not analyze entity-level information when establishing if a couple of mentions are referent or not. Therefore, beginning from this ascertainment it was suggest an entity-centric coreference resolution system, which composes coreference chains step by step. This system fits in the pattern stowing type as it engages a pipeline related architecture, where one stage has the grouped mention-pair patterns, which forebodes some rates, and the next stage holds the entity-centric pattern which utilizes previously engendered rates.

A. Coreference Resolution with Neural Networks

In the paper [4] the authors introduce a coreference resolution system using a neural network [5], whichever is capable to create distributed appearances of pairs of coreference groups. The appearances are used for the system to train when to merge groups. The system trains when to take alike decisions with the support of a simple first group-ranking procedure. This procedure classifies, in a downward sequence, the rates obtained from the group-ranking pattern with a design to own the highest rate entrant coreference links chiefly, and afterwards selects the ones with the highest rate. The group-ranking system is composed by a neural network, whose architecture (Fig. 1) has the subsequent subcomponents: mention-ranking pattern, mention-pair encoder, group-pair encoder, and group-ranking pattern.

II. THEORETICAL FOUNDATION AND ANALYSIS OF THE EXISTING SYSTEMS

A. Related Work

Viewed as a closed ensemble, the primary system can be considered as just an application which, plighted as input an English sentence from the statement of a geometry problem [6], releases as output an Abstract Meaning Representation graph (Fig. 2). The available solution can establish only in some measure the wanted output, signifying it can just build an AMR tree (AMR graph without coreferences) for the chosen sentence. Nevertheless, it is not still possible to mark the AMR nodes and relations with labels. Notwithstanding this model can emerge to be rather humdrum in the situation when considered the optimum solutions have under 70.0 out of 100.0 smatch f-scores from a simplistic view [7]. This implies that the AMR area also affords a large possibility for investigation and development.

Figure 1. Neural network coreference resolution architecture

DOI reference number: 10.18293/SEKE2021-168
Analyzing in profoundness, the primer solution is a transition-based parsing system, whichever enforces a set of transitions to the input sentence with a design to afford its equivalent AMR graph. The transitions appertain to every of the primary states of the system, beginning from the primary state till the final one, are trained with a Stack Long Short-Term Memory neural network [8]. Thereby, utilizing the trained model engendered from the LSTM neural network, the sequence of transitions can be automatically foreshadow starting from a selected geometry sentence. AMR graph [2] is a rooted, directed, acyclic graph, \( G = (V, A) \), whose edges and leaves are marked. The starting solution transition system includes the fundamental items of an arc-standard parser [9]: a stack (S), a buffer (B) and a group of activities, whichever are applique to the items of the stack and buffer with the purpose to shift by one parser shape to another. A remarkable script involves that the ending state has a blank buffer, a stack holding just one node, which is the root of the AMR graph, and the collection of applique activities on each transition [10]. The parser is trained on data collections that just include English sentences from the statement of a geometry problem and their equivalent Abstract Meaning Representation graphs.

B. Preprocessing, Learning, Postprocessing

The primary system utilizes a preprocessing phase to refers several from the attributes of the Abstract Meaning Representation area and create various input models parsed. Similar attributes are presented by Named Entities, Date Entities, quantities, or have-org-roles. The preprocessing phase rental consists in simplification of the learning models with the purpose for the learning method [11] to be qualified to generalize in the best possible way. The learning method underlying on the Stack Long Short-Term Memory architecture, as depicted in [12], and it constitutes the start system's solution, to the tough assignment of transition-based dependency parsing. It utilizes three distinct LSTMs: one for the buffer, which keeps the tokens of the sentence; one to the stack, which includes the AMR nodes; and one for the anterior foreshadowed activities [13]. The postprocessing phase signifies, in a large mode, the opposite of the preprocessing phase, which supplants Named Entities, Date Entities, quantities, have-org-roles, and is constituted from two important activities: subtree reattachment and AMR graph build from the foreshadowed sequence of actions. Furthermore, this restriction is emphasized by the incidence of sentences which include coreferences from our data collections: each data collection comprises approximately 50% of data which includes coreferences [14]. Precise coreferences [15] are referring entities which are contain in the sentence. The precedent selected situations all had clear co-references, from all terms, which referenced a concrete entity, presented in the sentence, and were not deducted.

III. DESIGN AND IMPLEMENTATION

The ascensive data-processing application comprised the next modules: data extraction, preprocessing, action sequence generation, training, postprocessing and evaluation. Data extraction analyzes the input files and takes out the pairs compound from sentence and gold regular AMR graph. The utilized data collections contain files where are data which are split into training, testing and development. They include AMR graphs, their equivalent sentences, and some information such as alignment, record id and annotator [16]. Fully, there are nine collections of data and they were extracted from different sources, such as forums, journals, and geometry books. Compound, they include 18946 items, which are divided amongst the training collection (16104), test collection (1515) and development (1327).

The chosen programming language to develop this system is Python. The main reasons why Python was chosen are accessibility, and scientific and numeric computing. Python is one of the most widely used programming languages for data science and data statistics. Another tool used was spaCy, an open-source library for Natural Language Processing (NLP), written in Python and Cython. It was built to aid in the construction of information extraction or Natural language Understanding systems, or to preprocess text for machine learning and deep learning. NeuralCoref [17] is a neural network pattern for annotating and solving coreference groups and represent an extent for spaCy. It is production-ready and can be enlarged to novel training data collections. NeuralCoref is an adjustment of a mention-pair pattern.
IV. TESTING AND VALIDATION

A. Evaluation

The assessment of the system is accomplished on the ensued AMR graphs from the input attempt models. They are resembled towards the Gold regular AMR graphs and a likeness grade is achieved by this procedure. Inasmuch as the AMR area is established on semantic parsing, this implies that exist more AMR graphs which are correlate with a selected sentence. Whereas the interpretations of the AMR graphs are alike, their compositions are, in some extent, various by one to another. Because there is no one exclusive text-to-AMR conversion, the assessment side is sooner complicated. The metric used by the Natural Language Processing collectivity is one which estimates the degree of semantic overlap [17]. The conclusive grade of the system is calculated as an f1 grade over the entities, starting by the foreshadowed collection, which fitted their equivalent gold regular AMR graphs [18].

The f-measure is calculated just once, at the final, afterwards every fit elements and unfit elements were appended simultaneously. The f1 grade is calculated at a great level with the purpose not to distinguish amongst the measures of the AMR graph. Besides from the smatch and f1 grade, it was likewise computed the accuracy of the system as an assessment metric. The accuracy is calculated on the foreshadowed activities reported to the gold regular actions foreshadowed by the Action Sequence Generation pattern. This metric likewise has an analogical deficiency because manifold AMRs can be correlate with an especial sentence. There are distinguished arrangements of the activities which could build the identical AMR graph. Therefore, for assessment, it was utilized the further metrics: accuracy and smatch f-grade. For computing the smatch f-grade were utilized the further three characteristics, foreshadowed by the system starting by the test items: M - the complete number of (parent, relation, child) trios that fitted into the gold regular side of the AMR, T - the complete number of (parent, relation, child) trios foreshadowed through the system and G – the complete number of (parent, relation, child) trios through the gold regular AMR. The accuracy symbolizes the fragment of right activity sequence predictions build from the system towards the gold regular one. If every foreshadowed activity sequence fit in the gold regular activity sequences, the accuracy is 1.0, else, if none of the foreshadowed activity sequences fit the gold regular one, the accuracy is 0.

B. Results and Comparison with Primary System

The purpose of this work was to increase the available implementation of the parser. In this scope, it was attempted to manipulate the principal restriction of the existing solution, the reality that the existing system could not operate using coreferences. The collection that the pattern was trained on includes umpteen coreferences and, thanks to this fact, and its other restrictions, the existing system could embrace benefit of just 23.85% of the used information. The boarding utilized was to manipulate coreferences in the preprocessing stage, which would ensuing in felling the parser with much divers training items. Thereby, if the parser could be trained on much diverse situations, afterwards, the pattern would be capable to generalize in a superior way and meliorate its capability to foreshadow action sequences. The principal argument wherefore the existing system could parse not many items was since in the collection occur many coreferences. By utilizing the NeuralCoref instrument in this solution, it was identified a fine proportion of the available precise coreferences, attendant in the collection. For every sentence in which precise coreferences were identified, it was enforced to the AMR graph to tree conversion method. The outcomes of this process can be viewed in Table 1. Besides the 4588 sentences in which we established coreferences and enforced the AMR conversion method, the Action Sequence Generation pattern was capable to calculate action sequences for 630 of them, thus, we generated 630 novel parsable items to supply the system. Since the 4588 sentences with their equivalent AMR graphs, 2219 from them were threw in the preprocessing phase from a certain of the four patterns of the existing solution: nominated entity substitute, data entity substitute or sizes substitute. Accordingly, to the existing 4809 parsable items on which the existing system was trained, were appended another 630 items, which signified a 13.3% expand in the input information. To calculate the ending outcomes, it was beginning at the existing design of the system and learned it manifold variants by modifying the value of epochs.

The activity sequence size and the word encapsulated scales stand the same from 25 sequence size and from 200 word size. Because the learning side uses so much time consuming, the system was just instructed for next three epoch types: 15, 20 and 25 epochs. In the Table 2 it can notice the outcomes of the system less the coreference manipulation. Established in advance, the metrics utilized to assess the system are smatch f-grade and the accuracy. Resembling the primer outcomes with that which have incorporated the coreference manipulating, it can observe that there is nearly a 5% growing in the smatch f-grade and a 4% growing for the accuracy. It is simple to notice that these ameliorations are firmly bound to the 630 novel training examples generated by the coreference manipulation solution.

<table>
<thead>
<tr>
<th>Models</th>
<th>Coreferences</th>
<th>Identified coreferences</th>
<th>Preprocessing models threwed</th>
<th>Primary parsable models</th>
<th>Novel parsable models</th>
</tr>
</thead>
<tbody>
<tr>
<td>926</td>
<td>540</td>
<td>178</td>
<td>153</td>
<td>179</td>
<td>19</td>
</tr>
<tr>
<td>1327</td>
<td>886</td>
<td>505</td>
<td>142</td>
<td>202</td>
<td>32</td>
</tr>
<tr>
<td>214</td>
<td>78</td>
<td>29</td>
<td>10</td>
<td>84</td>
<td>7</td>
</tr>
<tr>
<td>7281</td>
<td>3352</td>
<td>1150</td>
<td>1017</td>
<td>2057</td>
<td>229</td>
</tr>
<tr>
<td>689</td>
<td>111</td>
<td>45</td>
<td>11</td>
<td>394</td>
<td>28</td>
</tr>
<tr>
<td>6894</td>
<td>3163</td>
<td>2273</td>
<td>714</td>
<td>1445</td>
<td>259</td>
</tr>
<tr>
<td>204</td>
<td>140</td>
<td>76</td>
<td>31</td>
<td>22</td>
<td>8</td>
</tr>
<tr>
<td>866</td>
<td>484</td>
<td>253</td>
<td>79</td>
<td>216</td>
<td>30</td>
</tr>
<tr>
<td>345</td>
<td>86</td>
<td>48</td>
<td>39</td>
<td>156</td>
<td>11</td>
</tr>
<tr>
<td>18946</td>
<td>8930</td>
<td>4588</td>
<td>2219</td>
<td>4809</td>
<td>630</td>
</tr>
</tbody>
</table>
TABLE II. TRAINING OUTCOMES FOR 15, 20 AND 25 EPOCHS

<table>
<thead>
<tr>
<th>Type</th>
<th>Epochs</th>
<th>Sequence size</th>
<th>Embedding size</th>
<th>Smatch f-grade</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without coreference</td>
<td>15</td>
<td>25</td>
<td>200</td>
<td>38.97</td>
<td>88.99</td>
</tr>
<tr>
<td>With coreference</td>
<td>15</td>
<td>25</td>
<td>200</td>
<td>42.87</td>
<td>92.64</td>
</tr>
<tr>
<td>Without coreference</td>
<td>20</td>
<td>25</td>
<td>200</td>
<td>39.78</td>
<td>89.76</td>
</tr>
<tr>
<td>With coreference</td>
<td>20</td>
<td>25</td>
<td>200</td>
<td>44.36</td>
<td>93.07</td>
</tr>
<tr>
<td>Without coreference</td>
<td>25</td>
<td>25</td>
<td>200</td>
<td>40.02</td>
<td>90.02</td>
</tr>
<tr>
<td>With coreference</td>
<td>25</td>
<td>25</td>
<td>200</td>
<td>44.97</td>
<td>93.67</td>
</tr>
</tbody>
</table>

Like a remark, the significant distinction among the accuracy and the smatch f-grade it is a consequence of the situation that the mean value for sequence size is 10, thereby, for the network is more convenient to learn from the ultimate side of the action, conversely of learning the complicity of the sequences of the parsing stages. In this way just grows the accuracy and hold no effect on smatch, because the smatch just handles by AMR graphs.

V. CONCLUSIONS

In this paper it was tried to improve the main limitation of the primary system by coreferences manipulation. This goal has been achieved in the preprocessing and postprocessing stages of the system pipeline since replacing the parser would have implicated beginning a whole novel project from scheme, and not ameliorating the being one. The contribution for ameliorating the system to manipulate coreferences can be divided in two major stages. The first stage has the purpose to discover an algorithm for detection of explicit coreferences in the input sentences. For this purpose, it was used a tool named NeuralCoref, which utilizes a mention pair pattern and two neural networks for approximating grades. Starting from these, two tokens are reported as being coreferent or not. It has identified approximately 50% of the available coreferences from the collections. It desires emphasized the situation that it was identified just portion of the explicit coreferences existent in a data collection which included both implicit and explicit coreferences.

The succeeding stage was to encapsulate into the AMR parser the reveals from the above stage. In this scope, it was designed a method which would convert an AMR graph within an AMR tree, whencesoever precise coreferences were establish in the input sentence. Through this embedding, it was succeeded to engender 630 novel training models to the 4588 being ones, as can be seen in Table 1. This 13.73% growing in entrainment models aided the system to generalize in a better percentage and ensued in a 5% growth of the universal smatch f-grade and accuracy resembled to the primary system outcomes as viewed in Table 2.

REFERENCES

AutoCom: Automatic Comment Generation for C Code

Zhikang Tian
Shenzhen Foreign Language School
China

Yuekang Li
Nanyang Technological University
Singapore

ABSTRACT
Code comments improve program comprehension and program maintenance. However, the lack of comments is a common problem in industry. It is time- and manpower-consuming to add comments for large code bases. Thus, it is desirable to develop techniques for automatic comment generation. Previous works for automatic comment generation use deep learning or machine learning techniques. These techniques require a large amount of training data which is often unavailable or hard to acquire. This paper proposes a light-weight approach called AutoCom for automatic comment generation. In AutoCom, we first analyze the source code to extract key information. Then, we use the extracted information to search for appropriate text from a large programming Question and Answer (Q&A) site. Lastly, we use NLP techniques to convert the search result into code comments. In addition to the generated code comment, we also add predefined comments for library function usage in the source code. With AutoCom as the back-end, we built a web service which allows the user to upload source code and get it commented.

KEYWORDS
automated comment generation; documentation; program comprehension

1 INTRODUCTION
Comments play a very important role in code comprehension in program development and maintenance, especially after codes are created by programmers. As it is common for developers to shift between projects or departments, codes without comments would be hard to modify by programmers who take over. In the meantime, regularly commenting the codes would be distracting for programmers, as writing comments will break the train of thought of writing codes. Thus, the efficiency of the programmers might be brought down, and the codes written not fluent or even not functioning well. If the comments can be written automatically, it will save the programmers from writing comments and also help the successors in understanding previous codes. This thought leads to the birth of AutoCom, which is a program that can automatically match the codes with useful comments.

Existing automatic comment generation methods are not flawless: Most of them mainly focus on deep learning and machine learning. These techniques require a large amount of training data to fully function, but the data are hard to acquire. Thus, human labor is still not fully emancipated from the onerous process, because the large number of training data must be written by programmers based on their experience. Besides it also requires large amount of time to get these data. Secondly, those training data are complex, and it is hard to guarantee that no error has been made when writing. If any errors exist in the data, the result of commented codes will be wrong and useless. Thirdly, the large amount of new training data are continuously increasing, and they need to have a large database to save them, which requires money and space.

To address these issues, we proposed a new approach — AutoCom, a light-weight and full functioning tool in automatic C code comment generation. In AutoCom, we take down all the source code apart and extract out the key information of the source code. Then, we use the extracted key information to form the appropriate key words or filters and further establish a search on a large Question and Answer (Q&A) Site StackOverflow, which contains large number of source code and code description written by programmers. These Q&A work as the source to extract comments. After this, we analyze these sentences using Natural Language Processing (NLP) techniques, giving each sentence a weighting. We also convert the search results into useful comments. Lastly, we insert the comments into the source code. Additionally, we also add predefined comments for C Standard Library function usages in the source code, which requires fewer source code to be searched on StackOverflow. This novel design can save a lot of time when generating codes.

We implemented AutoCom as a Python-based commandline tool and wrapped it with a web-server to provide service: http://tianzhikang.pythonanywhere.com/. We also prepared an introduction video for AutoCom at: https://youtu.be/jxP389kFb7U.

2 METHODOLOGY & IMPLEMENTATION
In this section, we introduce details about the design of AutoCom. Figure 1 shows an overview of AutoCom's workflow. The input is C program source code and the output is the source code with comments added. The workflow of AutoCom has four steps:

1. We need to extract key information from the source code. To do so, we first convert the source code into abstract syntax tree (AST) to identify the different components. Then we extract function names and constant strings as individual pieces of key information from the AST.

2. We conduct a fuzzy search on the Q&A site for each piece of extracted information. After searching, we merge all the searching results (questions on the website) and rank them according to the sum of vote up/down scores and number of answers. More up votes and answers lead to a higher rank, and then the searching results with ranks higher than a threshold (the threshold can be specified by the user) will be kept and used to generate comments. Algorithm 1 shows the how the search results are filtered and retained. Note that here we discard the results of the keywords with more than 30 search results. The rationale is that if a keyword produces too many search results, it means the keyword is too general to represent the features of the code. For example, the function name "main" is a bad keyword for searching with. Here, we set 30 as the threshold after some experiments.
After the most suitable search results have been recognized and retained, we then apply text summary techniques to generate the summaries of the searched text. In AutoCom, we allow the users to choose either Latent Semantic Analysis (LSA) summarizer [8] or Edmundson Heuristic Method (EHM) summarizer [9]. For LSA, users only need to input the original text as well as a list of stop words to get the summary. For EHM, users need to provide two additional lists of bonus words and stigma words. In EHM, the additional words (cue phrases, keywords, title words) will increase the weight of a sentence while the stigma words shall decrease it. As for the additional words, we reuse the keywords to search the results. This means that sentence containing the keywords for searching are more likely to be included into the summary. As for the stigma words, we provide a list of words which are commonly used in the Q&A context of the website, such as what, why, error, warn, etc. This ensures that the commonly used sentences related to the question asking/answers or descriptions of the errors are excluded from the summary.

Last but not least, the generated summaries are inserted back into the code as comments. The summary will serve as a comment for the entire code. During this process, we also add predefined comments for standard C library function calls. This is achieved by maintaining a dictionary of standard library functions and their descriptions and by adding the corresponding descriptions as comments for each occurrence of the library function call in the code.

The key logic of AutoCom is implemented with around 400 lines of Python code. We use Py-StackExchange [6] to query the StackOverflow API. We use Beautiful Soup [3] for parsing the Q&A webpage and extracting the text content. We use pycparser [7] to parse the C code and produce the abstract syntax tree of the input code. Lastly, we use nltk [5] and sumy [2] to summarize the text.

Moreover, we wrap the key logic of AutoCom with a webserver based on Django [4]. In other words, we make AutoCom a web-service so that everyone can try and use it. AutoCom is currently hosted on PythonAnywhere [1]:http://tianzhikang.pythonanywhere.com/.

### 3 FUTURE WORK

In the future, we have three directions to improve the performance of AutoCom. First, we can improve the quality of information extraction from the source code. Currently we are using syntactic information like function names or constant strings as the keywords for searching. First, we plan to take a further step to extract some basic semantic understanding about the code. Second, we plan to apply finegrained analysis for the search result. Currently we are extracting only the text of the question. We can also extract the code fragments from the question if the question contains some. Then we can analyze the semantic of the extracted code fragment to see if it matches our original source code. If the code fragment matches our original source code, it means the question is suitable to generate comments. Third, we can use more advanced NLP techniques to generate comments. Currently, we are using traditional text summarization methods like LSA or EHM. In the future, we are planning to adopt more advanced text summarization techniques [10–12] in AutoCom.

### REFERENCES

Water-Wheel: Real-Time Storage with High Throughput and Scalability for Big Data Streams

Yanqi Lv, Ruicheng Liu, Peiquan Jin

1School of Computer Science and Technology, University of Science and Technology of China, Hefei, China
2Key Lab. of Electromagnetic Space Information, Chinese Academy of Sciences, Hefei, China
jpq@ustc.edu.com

Abstract—In this demonstration, we present a real-time storage system called Water-Wheel for big data streams. Big data streams involve rapid and continuous data flows, which will overwhelm the bandwidth capacity of conventional disk storage systems. The new Water-Wheel system proposes a new storage model that employs multiple nodes (servers) to accept data flows using a rotation way, like a traditional water wheel. When a node becomes full, it is transformed into a data-dumpling state that will flush the data in the node to durable storage, after which the node will become empty again to accept new data flows. We demonstrate that the Water-Wheel system can offer higher and more scalable storage bandwidth than existing big data storage systems that use a master-slave architecture to handle big data streams. After an introduction to the architecture and key designs of Water-Wheel, we present a case study to demonstrate Water-Wheel.

Keywords—Big data, Real-time storage, Throughput, Scalability

I. INTRODUCTION

The rapid development of the Internet of Things (IoT) [1] leads to big data streams in many applications. For example, there will be over 100 millions data inputs every second in a smart-grid monitoring application. Such highly-arriving big data makes it difficult to persist data in the data center using existing approaches [2], [3].

In this paper, inspired by the ancient water wheel in China, which can transport water from one spot to another through some rotating buckets, we propose a new real-time storage system called Water-Wheel for big data streams. We devise a rotation storage model [4] that can distribute data flows among multiple data nodes. This approach can make full use of each node in the data center and maximize the write throughput of the data center. Briefly, the Water-Wheel system has the following unique features:

(1) **High Throughput.** It employs a rotation storage model to deal with the conflict between highly-arriving data stream and low write throughput of the underlying storage. According to the rotation storage model, we set the memory of each node as a data bucket and all data buckets are rotated from the state of idle waiting to data filling, write waiting, and data dumping. With this mechanism, we can use a few data buckets to meet the high-throughput need required by big data streams.

(2) **High Scalability.** Water-Wheel is implemented based on the share-nothing architecture. New data nodes can be easily appended to the system and its storage capacity will be automatically added to the system. In case of increasing data flows, Water-Wheel offers high scalability by adding nodes seamlessly.

II. ARCHITECTURE OF WATER-WHEEL

The key design of Water-Wheel is a rotation storage model [4], which consists of a set of data buckets. A data bucket corresponds to a buffer in a data node in a cluster. All data nodes form a huge storage space that can be regarded as a ring because each data node is used with a round-robin manner.

All the buckets work according to the process indicated by arrows in Fig. 1. We can see that the Water-Wheel works in memory and the underlying storage nodes provide persistent storage service. When a bucket is filled in memory, it will be persisted to some storage node. We manage the states of all buckets and develop a state-transition scheme to make each bucket work at a right manner. As shown in Fig. 1, each bucket has one specific state at every moment. There are four states designed for buckets:

- State 1: **Idle Waiting.**
- State 2: **Data Filling.**
- State 3: **Write Waiting.**
- State 4: **Data Dumpling.**

Note that each bucket changes its state according to the given sequence, i.e., from State 1, 2, 3, and 4. Then, it will repeat the state transition from State 1 to 4. With such a mechanism, if one bucket becomes full, it will be changed into the state of Data Dumpling. And after we write the bucket to persistent storage, we can reuse the bucket, meaning that we can put the bucket into the waiting queue and let it wait to accept new data insertions. This is done by setting the bucket’s state to **Idle Waiting**.

Water-Wheel is deployed as a middle layer between the data stream and the underlying storage node. Combined with the state transition of data buckets, we give the general workflow of Water-Wheel in Fig. 2:

(1) When the data stream arrives, Water Wheel will first save the data in the data buckets in the state of data filling. Note that the number of the data buckets in the state of data filling can be configured in advance. Generally, more data-filling buckets can accept a bigger data stream. However, it will also incur higher pressure when dumping the data from
memory to persistent storage. We set the number of data-filling bucket to one by default.

(2) The a data-filling bucket becomes full, we change its state from data filling to write waiting to let the bucket not accept newly-arriving data streams.

(3) all the buckets in the state of write waiting will be dispatched to some storage nodes, which will be responsible for moving the data from memory to persistent storage.

(4) When a storage node gets a bucket in the state of write waiting, it will change the state of the bucket into data dumpling. Then, the storage node starts to move the data in the write-waiting bucket to files in persistent storage, e.g., SSDs or HDDs.

(5) After a storage node has completed the data dumpling of a bucket, which means that all the data in the bucket has been persisted, it will change the bucket’s state from data dumpling into idle waiting. And all idle-waiting buckets are organized into a queue. When newly-arriving data streams come to the system, we will select one ore more idle-waiting buckets from the queue to accept data streams. Meanwhile, those selected buckets will be marked as the state of data filling.

The above process will be executed repeatedly when the data stream comes to the system continuously. As all data buckets are reused to accept newly-inserted data, we can infer that the system can deliver a high write throughput. Even when the data stream becomes extremely fast, e.g., more than 10GB data written in one second, we can configure more data-filling buckets and let each bucket be equipped with large memory (e.g., using persistent memory [5], [6]), so that the system can absorb more than 10GB data in one second.

III. Demonstration

We implemented Water-Wheel on top of MongoDB [3]. We designed a graphical user interface for Water-Wheel, as shown in Fig. 3. User can monitor the real-time write throughput of the system. Users can also monitor the state of each node, such as the CPU utilization and write throughput. Water Wheel provides high scalability, meaning that we can easily add a new node into the system. Figure 4 shows the interface of adding a new node to the system. In the future, we will deploy our system to a cloud storage platform, so that it can support experiments running on hundreds of nodes.

ACKNOWLEDGEMENTS

This work was supported by the National Science Foundation of China (62072419). Peiquan Jin is the corresponding author.

REFERENCES

MT4ImgRec: A Metamorphic Testing Tool for Image Recognition Software

Dongyu Cao, Hongjing Guo, Chuanqi Tao

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
Ministry Key Laboratory for Safety-Critical Software Development and Verification, Nanjing University of Aeronautics and Astronautics, Nanjing, China

Correspondence to: taochuanqi@nuaa.edu.cn

Abstract—Although data-driven image recognition software has widely emerged in various fields, they suffer from quality issues. Metamorphic testing has been successfully applied to AI software for alleviating test oracle problems. Nevertheless, metamorphic testing still relies on manual methods in most cases, which is time-consuming. To improve test efficiency, a testing tool called MT4ImgRec is designed to automatically perform metamorphic testing for image recognition software.

Keywords: metamorphic testing; image recognition testing; tool

I. INTRODUCTION

The major challenge in testing image recognition software lies in the lack of a test oracle, which checks the correctness of the test execution [1]. Metamorphic testing is an effective manner that generates new test cases using the MRs (Metamorphic Relations) and determines whether the test passes by verifying whether the MRs are meet [2]. However, metamorphic testing is mostly performed manually. Thus, this paper implemented MT4ImgRec, a testing tool of image recognition software using metamorphic testing, which reduces the expenses in the test and enhances the test efficiency.

II. FUNCTION MODULES

MT4ImgRec consists of five modules. Figure 1 shows the interface of MT4ImgRec. The current version could be found in https://github.com/Miracy/MT4ImgRec.

Figure 1. Interface of MT4ImgRec

A. Original Test Data Management Module

After receiving the path of the test case folder, the module checks the original test cases for format and validity one by one. If passing the inspection, its file information is recorded.

B. Test Data Augmentation Module

The test data augmentation module aims to amplify the original test data by using segmentation mutation and adding the rainy effect. Segmentation mutation is realized by a YOLACT algorithm [3], which can quickly detect and divide the part of the target object of an image. Adding rainy effect is achieved by generating different densities of random noise, and elongating and rotating them.

C. Test Execution Module

For the original test data after amplification, the corresponding MRs are selected to automatically generate follow-up test data. MT4ImgRec implements six MRs for the image recognition software, including brightness, rotation, translation, cropping, scale, and blurring (averaging, gaussian, median, and bilateral). After generating follow-up test data, MT4ImgRec runs the program with a pair of test inputs (the original image and follow-up image) and records the output recognition result for each image.

D. Test Oracle Checking Module

The test oracle checking module judges the metamorphic test results by comparing the execution results of the original and follow-up test data. If two results are consistent, the metamorphic test result is displayed as passed, and the converse is not passed.

E. Test Result Analysis Module

The test result analysis module sorts and analyzes the test results. Different MRs and the number of test error cases of each parameter will be displayed. This tool also visualizes test results by using bar charts, including every MR-error recognition ratio and every parameter of MR-error recognition ratio.

III. CONCLUSION

MT4ImgRec is designed to test image recognition software using metamorphic testing, which can automate repetitive testing activities to enhance test efficiency. In the future, we will develop the testing tool for other intelligent software.

ACKNOWLEDGMENT

This work is supported by the Foundation of Graduate Innovation Center in Nanjing University of Aeronautics and Astronautics (kfjj20201603).

REFERENCES


DOI reference number: 10.18293/SEKE2021-205
Post2Event: Extracting Key Events from Microblogs

Chongwei Wang a, Xujian Zhao a*, Peiquan Jin b, Hui Zhang b, Chunming Yang a, Bo Li a

a Southwest University of Science and Technology, Mianyang 621000, China
b University of Science and Technology of China, Hefei 230027, China

* jasonzhaoxj@gmail.com

Abstract—This paper demonstrates a prototype system called Post2Event that aims to extract key events from microblogs. While many events are hidden in microblogs, people may only care about those critical events, which are named key events in Post2Event. Specially, we propose to model the topic-related significance of an event and integrate the influence with the temporal characteristics of the event to measure the event’s importance. We briefly present the architecture and technical details of Post2Event. Then, we report the comparative results of Post2Event on a real dataset. Finally, we demonstrate the running process of the system.

Index Terms—Event extraction, Social influence, Temporal distribution, Microblog

I. INTRODUCTION

Nowadays, social events occur every day and spread globally, forming a vast information source of social events. Meanwhile, news hotspots propagate through social networks and develop time-varying topics. This paper presents a prototype system called Post2Event to detect the key events in time-varying topics. Such a system can help users quickly understand the public situation on microblogging platforms.

Previous solutions to event extraction from microblogs relied on textual similarity, e.g., the TF-IDF similarity. However, the TF-IDF model always represents an event by some keywords and can only reflect the word-level textual importance of an event. Other graph-based approaches proposed to model events as graphs and use dominating set algorithm [1] or clustering coefficient information [2] of graph nodes to extract key events, but they failed to capture the time dimension of events, i.e., the importance of event may vary with time and evolution of topics.

Differing from existing approaches, We propose a new way to measure the importance of events. Specially, we propose to model the topic-related social influence of an event and integrate the influence with the temporal characteristics of the event to measure the event’s importance. Following this idea, we develop the Post2Event system and demonstrate that it has higher effectiveness of key-event detection than existing methods.

II. DESIGN AND IMPLEMENTATION OF POST2EVENT

The architecture of Post2Event is shown in Fig. 1. It consists of three modules. The Preprocessing module is to clean data and generate candidate event set. The Social Influence module is to model the significance of events, and the Temporal Distribution module is to model the temporal distribution of events and extract key events.

The input of the system is a set of microblogs, \( P = \{p_1, p_2, \ldots, p_i, \ldots, p_n\} \), and each \( p_i \) is a timestamped microblog. The output of the system is a time-ordered list of key events.

A. Social Influence

Key events need to be able to represent the important information of topics on different timestamps fully. For social networks, opinion leaders are more influential to information diffusion because they usually deliver more critical information than ordinary users. To this end, microblogs posted by opinion leaders are more likely to be key events. Therefore, the forwarding number, commenting number, and the total number of likes are used to calculate an event’s representativeness. Specifically, the social influence of an event can be represented by Eq. 1.

\[
SI(e_i) = \log(\alpha \cdot f_n + \beta \cdot c_n + \gamma \cdot l_n + \varepsilon)
\]  

Here, the numbers of forwards, comments, and likes are denoted as \( f_n, c_n \), and \( l_n \), respectively. The symbol \( \alpha, \beta, \gamma \) and \( \varepsilon \) are the corresponding weights, and \( \varepsilon \) refers to the Euler number.

B. Temporal Distribution

Time is one of the essential characteristics of events. By modeling the time characteristics of events, it can effectively reflect the evolution of events with time. For events happening simultaneously, users may have limited attention, which means that users usually focus on events with high social influence. Temporal expressions in an event are regularized by the normalizing algorithm [3] and we consider the social influence of an event in different timestamps to extract key events.

For an event sequence \( E = \{e_1, e_2, \ldots, e_i, \ldots, e_n\} \), we record the events at each timestamp, obtaining the recording series of events \( E_t = \{T_1: \{e_1, e_2, \ldots, e_{j-1}, e_j, \ldots, e_{j+1}, \ldots\}, T_2: \{e_1, e_2, \ldots, e_{j-1}, e_j, e_{j+1}, \ldots\}, \ldots\} \). The social influence \( IW(T_i) \) in \( T_i \) is defined as:

\[
IW(T_i) = \sum_{e_i} SI(e_i)
\]
The number of events extracted at timestamp $T_i$ is defined as:

$$N(T_i) = \left(\frac{IW(T_i) - Min_IW}{Max_IW - Min_IW}\right) + n$$

Here, $IW(T_i)$ is the sum of social influence in $T_i$, $Max_IW$ and $Min_IW$ represent the maximum and minimum of social influence at each timestamp, respectively. $n$ is a constant that indicates the maximum value of key events extracted from each timestamp. And the outermost bracket denotes a rounding down function.

### III. PERFORMANCE OF POST2EVENT

We evaluate the performance of Post2Event a real dataset collected from Sina Weibo (https://weibo.com). Three baseline algorithms are selected for comparison:

1. **TF-IDF** [4]. It selects the events with a high TF-IDF score as key events.
2. **MWDS** [1]. It constructs a graph representing the similarity among events and uses the Minimum-Weight Dominating Set to detect key events.
3. **DCCI** [2]. It is also a graph-based approach that is similar to MWDS but uses the Degree and Clustering Coefficient Information to detect key events.

We use ROUGE and Redundancy as the metrics [5]. The redundancy of an event is defined as the similarity of the two events that have the highest similarity in the key-event set, and the redundancy of the key-event set is the sum of the redundancy of each event.

The results are shown in Table 1, indicating that Post2Event achieves the best performance among all algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>ROUGE-1</th>
<th>ROUGE-L</th>
<th>Redundancy</th>
</tr>
</thead>
<tbody>
<tr>
<td>TF-IDF</td>
<td>0.82</td>
<td>0.47</td>
<td>39.20</td>
</tr>
<tr>
<td>MWDS</td>
<td>0.68</td>
<td>0.44</td>
<td>41.48</td>
</tr>
<tr>
<td>DCCI</td>
<td>0.68</td>
<td>0.43</td>
<td>44.36</td>
</tr>
<tr>
<td>Post2Event</td>
<td>0.83</td>
<td>0.70</td>
<td>27.81</td>
</tr>
</tbody>
</table>

### IV. DEMONSTRATION

The framework of key event extraction is a web application developed using Python 3.8 and Flask framework. In the demonstration, we will demonstrate the use of some critical functionalities of Post2Event, including Data View and Event View. Figure 2 shows a snapshot of Post2Event.

**Data View.** Figure 2(a) shows the Data View. It displays the local dataset by default and the popularity of some ongoing datasets crawled by our system. Users can explore the local datasets, check the events, and visualize each data's details in the interface. We show a preview of the ZTE Incident event in Fig. 2(a). Also, Post2Event provides a query interface for users to retrieve relevant events.

**Event View.** Figure 2(b) shows the Event View. When users select a dataset and click the "View" button, Post2Event will perform the event extraction and display the extracted key events. As shows in Fig. 2(b), Zone 1 shows the key events of "2020 U.S. President Election", Zone 2 shows the keywords of each key event, and Zone 3 displays the distribution of the keywords.

### ACKNOWLEDGEMENTS

This work is partially supported by the Humanities and Social Sciences Foundation of the Ministry of Education (No. 17YJCZH260), the National Science Foundation of China (No. 62072419), the Sichuan Science and Technology Program, China (2020YFS0057) and the CERNET Innovation Project (NGII20180403). Xujian Zhao is the corresponding author.

### REFERENCES

CASTR: Assisting Bug Report Assignment Recommender Creation

Disha Devaiya, John Anvik, Farjana Yeasmin Omee, Meher Bheree
Department of Mathematics and Computer Science
University of Lethbridge, Alberta, CANADA
E-mail: devaiya86@gmail.com, [john.anvik, omee, bheree]@uleth.ca

Issue tracking systems are used to make a software development process more manageable, especially for a geographically dispersed team. However, for each bug report, a decision-making process called bug report triage needs to be performed. A common bug report triage decision is the assigning of a developer to a specific bug report.

Bug report triage can take significant time and resources. Bug report assignment recommenders have been proposed for reducing the workload of a project member. However, creating a recommender is complex, as project members have to perform several steps such as data preparation and selection of a machine learning algorithm. Although previous work has sought to find specific answers for aspects of the assignment recommender creation process, to the best of our knowledge, only a few other works have examined assisting with this recommender creation process.

CASTR (Creation Assistant for Supporting Triage Recommenders) [1, 2] is a platform-independent multi-tier web application. It allows a project member to analyze the dataset using a graphical representation and also assists in configuring project-specific parameters for a machine learning algorithm. This demonstration shows how to use CASTR to create a bug report assignment recommender, which consists of the following steps:

1. Download the dataset from the project’s issue tracking repository. CASTR provides an interface that allows a user to set parameters, such as a date range and number of reports, for downloading a dataset using the Bugzilla REST endpoints. For the demonstration, we will be using a data set from a large open-source project, such as Firefox, PlasmaShell or LibreOffice.

2. Configure and create the recommender. Information about the collected dataset is displayed by the Configuration tab. This tab assists a user in performing data filtration by setting project-specific heuristics for report labelling and a minimum activity threshold value for which labels (i.e. developers) to recommend. Also, the user selects a machine learning algorithm (SVM, Multinomial Naïve Bayes, C4.5, or rule-based) to use for recommender creation. Finally, the user can select an algorithm to handle data imbalance (oversampling using SMOTE, undersampling using clusters, or manual oversampling). Once the user has set the recommender configuration parameters, the user clicks the “Create Recommender” button and is taken to the Analysis tab.

3. Analysis of recommender. The Analysis tab presents evaluation results for the created recommender. It displays the Top-1, Top-3, and Top-5 precision and recall values for a testing set of 100 randomly selected reports that are not used for training. Also, the Analysis tab displays an example of possible labels and predictions for some randomly selected bug reports from the testing set. A History panel allows a user to select a previous recommender configuration. The Confusion Matrix tab displays for each developer name, the distribution of correct and incorrectly predicted class values from the testing set.

4. Retrain the recommender. If needed, the user can then return to the Configuration tab, adjust the configuration parameters, and create a new recommender. This process continues until the user is either satisfied with the created recommender, or the user has determined that an assignment recommender cannot be created with a high enough accuracy to benefit the project.

References

## Authors

<table>
<thead>
<tr>
<th>Authors</th>
<th>Pages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abhijit Mondal</td>
<td>359</td>
</tr>
<tr>
<td>Adrián Bernal</td>
<td>211</td>
</tr>
<tr>
<td>Agnieszka Ławrynowicz</td>
<td>437</td>
</tr>
<tr>
<td>Aimin Yu</td>
<td>510</td>
</tr>
<tr>
<td>Alberto Núñez</td>
<td>211</td>
</tr>
<tr>
<td>Aless</td>
<td>550</td>
</tr>
<tr>
<td>Alex</td>
<td>195, 223, 429</td>
</tr>
<tr>
<td>Alireza Javadian Sabet</td>
<td>429</td>
</tr>
<tr>
<td>Alok Ratnaparkhi</td>
<td>98</td>
</tr>
<tr>
<td>Amal El Fallah Seghrouchni</td>
<td>21</td>
</tr>
<tr>
<td>Amani Ibrahim</td>
<td>303</td>
</tr>
<tr>
<td>Amit Kumar Mondal</td>
<td>7</td>
</tr>
<tr>
<td>Anca-Elena Iordan</td>
<td>628</td>
</tr>
<tr>
<td>André Sousa</td>
<td>217</td>
</tr>
<tr>
<td>Angelo Perkusich</td>
<td>223, 229, 235</td>
</tr>
<tr>
<td>Arnault Lapitre</td>
<td>291</td>
</tr>
<tr>
<td>Arndt Lüder</td>
<td>37</td>
</tr>
<tr>
<td>Arthur Freire</td>
<td>223</td>
</tr>
<tr>
<td>Assia Belbachir</td>
<td>21</td>
</tr>
<tr>
<td>Aya Zaki Ismail</td>
<td>303</td>
</tr>
<tr>
<td>Bai Wang</td>
<td>592</td>
</tr>
<tr>
<td>Banani Roy</td>
<td>7, 241</td>
</tr>
<tr>
<td>Bei Hua</td>
<td>616</td>
</tr>
<tr>
<td>Beichuan Zhang</td>
<td>315, 321</td>
</tr>
<tr>
<td>Bharavi Mishra</td>
<td>516</td>
</tr>
<tr>
<td>Bin Wu</td>
<td>592</td>
</tr>
<tr>
<td>Bin Zhou</td>
<td>417, 598</td>
</tr>
<tr>
<td>Bing Li</td>
<td>86</td>
</tr>
<tr>
<td>Binghua Song</td>
<td>79</td>
</tr>
<tr>
<td>Bixin Li</td>
<td>423</td>
</tr>
<tr>
<td>Bo Li</td>
<td>637</td>
</tr>
<tr>
<td>Bohao Wang</td>
<td>166</td>
</tr>
<tr>
<td>Boutheina Bannour</td>
<td>291</td>
</tr>
<tr>
<td>Bowen Du</td>
<td>160</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Page Numbers</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Carmine Gravino</td>
<td>550</td>
</tr>
<tr>
<td>Chang Liu</td>
<td>582</td>
</tr>
<tr>
<td>Chen Yang</td>
<td>57</td>
</tr>
<tr>
<td>Cheng Chen</td>
<td>622</td>
</tr>
<tr>
<td>Chi Yu</td>
<td>499</td>
</tr>
<tr>
<td>Chongwei Wang</td>
<td>637</td>
</tr>
<tr>
<td>Chuanqi Tao</td>
<td>560, 636</td>
</tr>
<tr>
<td>Chuantao Yin</td>
<td>451</td>
</tr>
<tr>
<td>Chunming Yang</td>
<td>637</td>
</tr>
<tr>
<td>Dan Li</td>
<td>15</td>
</tr>
<tr>
<td>Dan Meng</td>
<td>510</td>
</tr>
<tr>
<td>Danilo Santos</td>
<td>229</td>
</tr>
<tr>
<td>Danylllo Albuquerque</td>
<td>229</td>
</tr>
<tr>
<td>Daryl Stone</td>
<td>67</td>
</tr>
<tr>
<td>David Dampier</td>
<td>261</td>
</tr>
<tr>
<td>Dawid Wiśniewski</td>
<td>437</td>
</tr>
<tr>
<td>Deming Sheng</td>
<td>189, 399</td>
</tr>
<tr>
<td>Diego Avila</td>
<td>340</td>
</tr>
<tr>
<td>Dietmar Winkler</td>
<td>37</td>
</tr>
<tr>
<td>Dionysis Athanasopoulos</td>
<td>369</td>
</tr>
<tr>
<td>Disha Devaiya</td>
<td>271</td>
</tr>
<tr>
<td>Disha Thakarshibhai Devaiya</td>
<td>639</td>
</tr>
<tr>
<td>Dong Wang</td>
<td>183</td>
</tr>
<tr>
<td>Dong Yang</td>
<td>528</td>
</tr>
<tr>
<td>Dongrui Fan</td>
<td>387</td>
</tr>
<tr>
<td>Dongyu Cao</td>
<td>560, 636</td>
</tr>
<tr>
<td>Duong Dinh Tran</td>
<td>120, 126, 138</td>
</tr>
<tr>
<td>Elena Gómez-Martínez</td>
<td>199</td>
</tr>
<tr>
<td>er Nemirovskiy</td>
<td>429</td>
</tr>
<tr>
<td>Everton Guimaraes</td>
<td>229</td>
</tr>
<tr>
<td>Famei He</td>
<td>79</td>
</tr>
<tr>
<td>Fanshu Shang</td>
<td>487</td>
</tr>
<tr>
<td>Faouzi Mhamdi</td>
<td>346</td>
</tr>
<tr>
<td>Farjana Yeasmin Omee</td>
<td>271, 639</td>
</tr>
<tr>
<td>Fatma Outay</td>
<td>261</td>
</tr>
<tr>
<td>Franz Wotawa</td>
<td>610</td>
</tr>
<tr>
<td>A-2</td>
<td></td>
</tr>
</tbody>
</table>
Ganghua Zhang 447
Gelin Zhang 166
Gleyser Guimarães 229, 235
Guang Yang 499
Guangdong Bai 172, 249
Guillaume Giraud 291
Guisheng Fan 75
Guoai Xu 458
Guosheng Xu 458
H
Haiyang Wang 417
Haiying Sun 183
Hanzhong Zheng 45
Hao Chen 92
Haoyu Wang 249
Haroon Malik 261
Harshit Singhal 516
Hernán-Indibil De la Cruz 211
Hiroyuki Nakagawa 544
Hong Zhou 451
Hongjian Jiang 31, 108
Hongjing Guo 560, 636
Hongqin Zhang 102
Huan Wang 564
Huawei Cao 387
Hui Zhang 637
Huibiao Zhu 102, 154
Huiqun Yu 75
Hujun Shen 205
Hyggo Almeida 223, 229, 235
J
Jahnavi Jonnalagadda 309
Jamy Chahal 21
Jan Herzog 37
Jedrzej Potoniec 437
Jerry Gao 560
Jian Wang 86
Jiang Li 564
Jianhui Zheng 464
Jiankun Cao 458
Jianqi Shi 166, 532
Jianxia Chen 582, 586
Jiaqi Yin 102, 154
Jiaxin Liu 528
Jiaxin Zhu 249
Jie Deng 458
Jie Yan 67
Jiexiang Kang 183
Jin Lv 108
Jin Xu 160, 183
Jincheng Xu 570, 576
Jinfang Cai 622
Jing Liu 183
Jing Yang 487
Jingling Yuan 189, 399
Jinyang Liu 451
Jiwen Luo 381
John Castro 275, 281
John Anvik 271, 334, 639
John Grundy 303
João Faria 217
João Moreira 217
Jun Jiang 79
Junfeng Li 15
Junfeng Zhao 1
Junsheng Chang 381
Junwei Sun 592
Justin Kramer 45
K
Kang Shi 470
Kang Zeng 417, 598
Karina Villela 340
Kazuhiro Ogata 114, 120, 126, 138
Kazutoshi Sakakibara 114
Ke Ma 538
Kentaro Waki 138
Kevin Schneider 7
Kirthy Kolluri 297
Kristof Meixner 37
Kui Xiao 375
Kunyang Wang 451
Kyller Costa Gorgônio 235
Kyller Gorgônio 223
L
Lan Lin 51
Lawrence Chung 297
Lian Gu 528
Liang Ge 564
Liang Hao 51
Liangyu Chen 622
Lianwei Li 592
Liu Wang 249
Long Chen 411
Lorenz Klampfl 610
Lucineia Heloisa Thom 340
Luyao Liu 538
M
Maham Nazir 592
Mahdi Hashemi 309, 441
Manuel Neto 223
María-Emilia Cambronero 211
Masaki Nakamura 114
Mathilde Arnaud 291
Matteo Giovanni Rossi 429
Maxime Savary-Leblanc 327
Md Nurul Ahad Tawhid 255
Meher Bheree 271, 639
Men Huichao 481
Meng Fan 510
Meng Sun 132, 150, 172
Mengbo Xiong 79
Mersedeh Sadeghi 429
Miaomiao Zhang 160
Michele Risi 550
Miguel Marroyo 199
Mingyu Yan 387
Minyang Yi 75
Mirko Perkusich 223, 229, 235
Mohamed Abdelrazek 303
Mohamed Osama 303
Moumita Asad 255
Mourad Elloumi 346
<table>
<thead>
<tr>
<th>Name</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rubens dos Santos</td>
<td>340</td>
</tr>
<tr>
<td>Rui Feng</td>
<td>1</td>
</tr>
<tr>
<td>Ruicheng Liu</td>
<td>616, 634</td>
</tr>
<tr>
<td>Ruiqing Li</td>
<td>249</td>
</tr>
<tr>
<td>Ruizhi Xiao</td>
<td>92</td>
</tr>
<tr>
<td>Safia Kalwar</td>
<td>429</td>
</tr>
<tr>
<td>Sameer Kulkarni</td>
<td>15</td>
</tr>
<tr>
<td>Sebastien Gerard</td>
<td>327</td>
</tr>
<tr>
<td>Sergei Shcherban</td>
<td>57</td>
</tr>
<tr>
<td>Shaoze Chen</td>
<td>352</td>
</tr>
<tr>
<td>Shayla Bhuyan</td>
<td>334</td>
</tr>
<tr>
<td>Shenghua Zhong</td>
<td>315, 321</td>
</tr>
<tr>
<td>Shihab Shahriar Khan</td>
<td>255</td>
</tr>
<tr>
<td>Shihui Wang</td>
<td>375</td>
</tr>
<tr>
<td>Shikuo Chang</td>
<td>45</td>
</tr>
<tr>
<td>Shingo Ariwaka</td>
<td>544</td>
</tr>
<tr>
<td>Shingo Takada</td>
<td>522</td>
</tr>
<tr>
<td>Shirish Singh</td>
<td>516</td>
</tr>
<tr>
<td>Shirui Sheng</td>
<td>586</td>
</tr>
<tr>
<td>Shiyang Li</td>
<td>604</td>
</tr>
<tr>
<td>Shouhong Wan</td>
<td>25, 616</td>
</tr>
<tr>
<td>Shuai Li</td>
<td>285</td>
</tr>
<tr>
<td>Shubai Chen</td>
<td>405</td>
</tr>
<tr>
<td>Shuho Fujii</td>
<td>120</td>
</tr>
<tr>
<td>Shuo Wang</td>
<td>411</td>
</tr>
<tr>
<td>Shuo Yang</td>
<td>411</td>
</tr>
<tr>
<td>Shuyuan Jin</td>
<td>92</td>
</tr>
<tr>
<td>Sijun Tan</td>
<td>108</td>
</tr>
<tr>
<td>Silvia Acuña</td>
<td>199, 275, 281</td>
</tr>
<tr>
<td>Siqing Yang</td>
<td>205</td>
</tr>
<tr>
<td>Sivasurya Santhanam</td>
<td>265</td>
</tr>
<tr>
<td>Siyi Li</td>
<td>458</td>
</tr>
<tr>
<td>Song Wu</td>
<td>405</td>
</tr>
<tr>
<td>Sristy Sumana Nath</td>
<td>7, 241</td>
</tr>
<tr>
<td>Stefan Biffl</td>
<td>37</td>
</tr>
<tr>
<td>Su Wang</td>
<td>352</td>
</tr>
<tr>
<td>Swapna Gokhale</td>
<td>359</td>
</tr>
<tr>
<td>Sébastien Gerard</td>
<td>285</td>
</tr>
<tr>
<td>Takwa Kochbati</td>
<td>285</td>
</tr>
</tbody>
</table>
Tao Wang 381
Tatsuhiro Tsuchiya 544
Thet Wai Mon 120
Tian-Ming Bu 470
Tianyuan Hu 423
Tobias Hecking 265
Tom Hill 297
V
Valentín Valero 211
Vitória Mendes 195
W
Wang Botao 481
Wei Dong 528
Wei Xuan 387
Wei Zhang 487
Weidi Sun 172
Weifeng Xu 67
Weihang Su 249
Weiwei Xu 503
Weiyu Pan 538
Wenxin Hu 352
Wenyi Guo 592
Wenzhi Wang 510
X
Xavier Le Pallec 327
Xia Mao 532
Xiang Chen 499
Xiaochun Ye 387
Xiaofang Zhang 503
Xiaokun Luan 132
Xiaoping Che 447
Xiaoting Guo 381
Xiaoxiao Sun 205
Xiaoyan Zhang 451
Xin Song 417, 598
Xin Sun 51
Xin Zhang 189, 399
Xingguang Yang 75
Xinjun Mao 411
Xiyue Zhang 132
Xuan Wen 586
Xujian Zhao 637
Xuren Wang 79
Y
Yan Lin 458
Yang Yang 532
Yanhong Huang 166, 532
Yanlin Zhou 499
Yanqi Lv 634
Yating Zhang 528
Yi Liu 98
Yi-Chun Feng 150
Yifan Dai 622
Yinyuan Zhang 411
Yongjian Li 108
Yongping Luo 25
Yongqiang Liu 86
Yongxin Zhao 31, 108
Yu Chen 405, 582, 586
Yuan Yuan 405
Yuekang Li 632
Yujia Liu 417, 598
Yuke Ying 205
Yuki Moriguchi 522
Yuki Okura 114
Yuqi Shen 622
Yuteng Lu 150, 172
Yuyuan Zhang 31
Z
Zekun Zhang 86
Zengyang Li 57
Zeqing Qin 375
Zesong Wang 375
Zeyu Lu 532
Zhaole Chu 25
Zhe Hou 166
Zheng Wang 31
Zhengwei Jiang 79
Zhenyu Pan 423
Zhijiao Xiao 315, 321
Zhikang Tian 632
<table>
<thead>
<tr>
<th>Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zhimin Tang</td>
<td>387</td>
</tr>
<tr>
<td>Zhiming Hu</td>
<td>31</td>
</tr>
<tr>
<td>Zhiru Hou</td>
<td>154</td>
</tr>
<tr>
<td>Zihan Xiong</td>
<td>79</td>
</tr>
<tr>
<td>Zihui Wang</td>
<td>528</td>
</tr>
<tr>
<td>Zinat Ara</td>
<td>441</td>
</tr>
<tr>
<td>Ziqi Shuai</td>
<td>538</td>
</tr>
<tr>
<td>Zishan Li</td>
<td>160</td>
</tr>
</tbody>
</table>
## Keywords

**Keywords:**

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Pages:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract Meaning Representation</td>
<td>628</td>
</tr>
<tr>
<td>abstract syntax tree</td>
<td>503</td>
</tr>
<tr>
<td>activity suffix prediction</td>
<td>205</td>
</tr>
<tr>
<td>advanced persistent threat</td>
<td>67</td>
</tr>
<tr>
<td>Adversarial Robustness</td>
<td>160</td>
</tr>
<tr>
<td>Agile</td>
<td>223</td>
</tr>
<tr>
<td>Agile Software Development</td>
<td>223</td>
</tr>
<tr>
<td>AI software quality validation</td>
<td>560</td>
</tr>
<tr>
<td>algebraic specification language</td>
<td>120, 126</td>
</tr>
<tr>
<td>Alice&amp;Bob specifications</td>
<td>108</td>
</tr>
<tr>
<td>AMR Parser A</td>
<td>628</td>
</tr>
<tr>
<td>Anomaly detection</td>
<td>86</td>
</tr>
<tr>
<td>anti-patterns</td>
<td>98</td>
</tr>
<tr>
<td>APTER-SENT</td>
<td>79</td>
</tr>
<tr>
<td>ATT&amp;CK framework</td>
<td>67</td>
</tr>
<tr>
<td>Attention</td>
<td>604</td>
</tr>
<tr>
<td>attention mechanism</td>
<td>352, 475</td>
</tr>
<tr>
<td>Attention Mechanism</td>
<td>399</td>
</tr>
<tr>
<td>authentication protocol</td>
<td>120</td>
</tr>
<tr>
<td>automated mapping</td>
<td>429</td>
</tr>
<tr>
<td>Automatic Refactoring</td>
<td>1</td>
</tr>
<tr>
<td>auxiliary variable</td>
<td>126</td>
</tr>
<tr>
<td>axiology</td>
<td>63</td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Bayesian method</td>
<td>309</td>
</tr>
<tr>
<td>Bayesian Network</td>
<td>235</td>
</tr>
<tr>
<td>Bi-GRU</td>
<td>582</td>
</tr>
<tr>
<td>Big data</td>
<td>634</td>
</tr>
<tr>
<td>Bloom filter</td>
<td>616</td>
</tr>
<tr>
<td>Bounded Sampling Ratio</td>
<td>387</td>
</tr>
<tr>
<td>C</td>
<td></td>
</tr>
<tr>
<td>Camur</td>
<td>346</td>
</tr>
<tr>
<td>characteristic</td>
<td>15</td>
</tr>
<tr>
<td>Chatbots</td>
<td>275, 281</td>
</tr>
<tr>
<td>Chunking Representation</td>
<td>321</td>
</tr>
<tr>
<td>City Similarity</td>
<td>447</td>
</tr>
</tbody>
</table>
Classification 217
classoverlap 75
Climate anomalies 309
Cloud-native system 86
clustering 285
code representation 516
code review 7
Collaborative Filtering 315
collusion and plagiarism 365
community 487
compiler optimization 516
component 261, 285
Computation Tree Logic 183
conceptextraction 7
conceptual model 365
conceptual modeling 63
Consensus Protocol 150
Construct 223
convolutional neural network 475
Cross-level attention 405
CSP 102, 154
Cyber physical system 31
D
data protection regulation 199
deep convolutional neural network 441
Deep Learning 57, 92, 321, 503, 528, 554
Deep Neural Network 172
deep reinforcement learning 458
DeepAuto 172
defect detection 423
degree discount 393
Design Pattern Analysis 45
Developer Contribution 249
Developer reply 261
digital forensics 67
distance education 365
Distant supervision 604
Dolev-Yao 108
Dot-product 321
Drones 261
human-centered agile methodology 63
Hybrid Memory 25
I
image recognition 560
image recognition testing 636
IMPA 346
Model-Agnostic Explanations 570
Agile Software Development 235
anomaly detection 92
Architectural change 7
CKB 150
cloud computing 15
CNF 538
code clones 516
code search 528
code smell 503
concept extraction 381
concolic execution 510
COVID-19 249
crack detection 458
data mining 598
Deep cross-modal hashing 405
Deep learning 604
Dubbo 154
Evolving Graph Streams 387
Formal Verification 172
Go 255
Gradient-based Explanatory Methods 576
Human Aspects 229
Hybrid systems 31
ICN 102
influence maximization 393
Just-in-time software defect prediction 75
Knowledge Graph 411
Model Checking 183
MOOC 189, 399
multi-event prediction 417
Multiple solution 346
Murphi 108
programmable logic controller 532
Reluplex Algorithm 160
<table>
<thead>
<tr>
<th>Term</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>repository mining</td>
<td>265</td>
</tr>
<tr>
<td>Requirements engineering</td>
<td>303</td>
</tr>
<tr>
<td>Risk Management</td>
<td>217</td>
</tr>
<tr>
<td>smart contract</td>
<td>423</td>
</tr>
<tr>
<td>Software release notes</td>
<td>241</td>
</tr>
<tr>
<td>software vulnerabilities</td>
<td>98</td>
</tr>
<tr>
<td>Teamwork</td>
<td>223</td>
</tr>
<tr>
<td>Textsummarization</td>
<td>592</td>
</tr>
<tr>
<td>Transfer Learning</td>
<td>447</td>
</tr>
<tr>
<td>transformer</td>
<td>352</td>
</tr>
<tr>
<td>Tree Ensemble</td>
<td>166</td>
</tr>
<tr>
<td>Industry 4</td>
<td>37</td>
</tr>
<tr>
<td>Industry4</td>
<td>37</td>
</tr>
<tr>
<td>Informal notation</td>
<td>303</td>
</tr>
<tr>
<td>intelligent preprocessing decision system</td>
<td>458</td>
</tr>
<tr>
<td>interactive system</td>
<td>63</td>
</tr>
<tr>
<td>interplay</td>
<td>532</td>
</tr>
<tr>
<td>Interpretable Model</td>
<td>189</td>
</tr>
<tr>
<td>IoT Middleware</td>
<td>102</td>
</tr>
<tr>
<td>J</td>
<td>255</td>
</tr>
<tr>
<td>joint extraction model</td>
<td>352</td>
</tr>
<tr>
<td>K</td>
<td></td>
</tr>
<tr>
<td>K-Means clustering</td>
<td>75</td>
</tr>
<tr>
<td>keywords attention</td>
<td>582</td>
</tr>
<tr>
<td>knowledge graph</td>
<td>423</td>
</tr>
<tr>
<td>Knowledge Tracing</td>
<td>399</td>
</tr>
<tr>
<td>knowledge-aware attention</td>
<td>417</td>
</tr>
<tr>
<td>L</td>
<td></td>
</tr>
<tr>
<td>Lightweight Pipeline</td>
<td>189</td>
</tr>
<tr>
<td>linguistic similarity</td>
<td>429</td>
</tr>
<tr>
<td>Local Explanations</td>
<td>570</td>
</tr>
<tr>
<td>log analysis</td>
<td>92</td>
</tr>
<tr>
<td>LSM-tree</td>
<td>616</td>
</tr>
<tr>
<td>LSTM</td>
<td>205</td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>Machine Learning</td>
<td>217, 554</td>
</tr>
<tr>
<td>Machine learning</td>
<td>538</td>
</tr>
<tr>
<td>MapReduce</td>
<td>1</td>
</tr>
<tr>
<td>MCS protocol</td>
<td>138</td>
</tr>
<tr>
<td>Term</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Merit</td>
<td>346</td>
</tr>
<tr>
<td>meta-path</td>
<td>564</td>
</tr>
<tr>
<td>metamorphic testing</td>
<td>636</td>
</tr>
<tr>
<td>Micro-services</td>
<td>45</td>
</tr>
<tr>
<td>Microservice</td>
<td>86</td>
</tr>
<tr>
<td>Mixed query</td>
<td>616</td>
</tr>
<tr>
<td>Mobile App</td>
<td>261</td>
</tr>
<tr>
<td>Model Checking</td>
<td>150</td>
</tr>
<tr>
<td>Model Mutation Testing</td>
<td>554</td>
</tr>
<tr>
<td>Model verification</td>
<td>31</td>
</tr>
<tr>
<td>Modeling</td>
<td>102</td>
</tr>
<tr>
<td>Modelling</td>
<td>275</td>
</tr>
<tr>
<td>MOOC</td>
<td>451</td>
</tr>
<tr>
<td>Multi-Attention</td>
<td>528</td>
</tr>
<tr>
<td>multi-sensor fusion</td>
<td>475</td>
</tr>
<tr>
<td>Multiclass Classification</td>
<td>57</td>
</tr>
<tr>
<td>Multiple Part</td>
<td>346</td>
</tr>
<tr>
<td>mutual exclusion</td>
<td>138</td>
</tr>
<tr>
<td>mutual exclusion protocol</td>
<td>126</td>
</tr>
<tr>
<td>natural language processing</td>
<td>265, 285, 429</td>
</tr>
<tr>
<td>Natural Language Processing</td>
<td>628</td>
</tr>
<tr>
<td>natural language processing</td>
<td>564</td>
</tr>
<tr>
<td>network embedding</td>
<td>487</td>
</tr>
<tr>
<td>Neural Network</td>
<td>57</td>
</tr>
<tr>
<td>NLP</td>
<td>411</td>
</tr>
<tr>
<td>Non-Volatile Memory</td>
<td>25</td>
</tr>
<tr>
<td>NYT</td>
<td>352</td>
</tr>
<tr>
<td>online education resources</td>
<td>375</td>
</tr>
<tr>
<td>online learning</td>
<td>381, 451</td>
</tr>
<tr>
<td>online teaching and learning</td>
<td>365</td>
</tr>
<tr>
<td>Ontology</td>
<td>297, 429</td>
</tr>
<tr>
<td>Open Source Software</td>
<td>249</td>
</tr>
<tr>
<td>PageRank</td>
<td>393</td>
</tr>
<tr>
<td>PAT</td>
<td>102</td>
</tr>
<tr>
<td>patch testing</td>
<td>510</td>
</tr>
<tr>
<td>performanceserverless</td>
<td>15</td>
</tr>
<tr>
<td>personal data</td>
<td>199</td>
</tr>
<tr>
<td>Personality</td>
<td>229</td>
</tr>
<tr>
<td>Term</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>prediction</td>
<td>441</td>
</tr>
<tr>
<td>prerequisite relations</td>
<td>375</td>
</tr>
<tr>
<td>Pretrained LanguageModel</td>
<td>592</td>
</tr>
<tr>
<td>privacy requirements</td>
<td>199</td>
</tr>
<tr>
<td>Probabilistic forecast</td>
<td>309</td>
</tr>
<tr>
<td>Probe Machine</td>
<td>183</td>
</tr>
<tr>
<td>process discovery</td>
<td>205</td>
</tr>
<tr>
<td>Production Systems Engineering</td>
<td>37</td>
</tr>
<tr>
<td>Program comprehensibility</td>
<td>255</td>
</tr>
<tr>
<td>program invariant</td>
<td>532</td>
</tr>
<tr>
<td>Program Transformation</td>
<td>1</td>
</tr>
<tr>
<td>proof assistant</td>
<td>120</td>
</tr>
<tr>
<td>proof generator</td>
<td>120</td>
</tr>
<tr>
<td>proof score</td>
<td>126</td>
</tr>
<tr>
<td>propagation probability</td>
<td>393</td>
</tr>
<tr>
<td>Property Verification</td>
<td>166</td>
</tr>
<tr>
<td>Psychometric Instruments</td>
<td>229</td>
</tr>
<tr>
<td>Python</td>
<td>628</td>
</tr>
<tr>
<td>Q</td>
<td></td>
</tr>
<tr>
<td>Quality Attributes</td>
<td>554</td>
</tr>
<tr>
<td>QuestionnaireSurvey</td>
<td>235</td>
</tr>
<tr>
<td>Queuing Network Modelling</td>
<td>45</td>
</tr>
<tr>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Real-time storage</td>
<td>634</td>
</tr>
<tr>
<td>real-time system</td>
<td>532</td>
</tr>
<tr>
<td>Recommender System</td>
<td>315, 321</td>
</tr>
<tr>
<td>recommender system</td>
<td>451</td>
</tr>
<tr>
<td>recoverable</td>
<td>138</td>
</tr>
<tr>
<td>Recurrent Neural Network</td>
<td>315</td>
</tr>
<tr>
<td>recurrent neural network demand travel learning</td>
<td>441</td>
</tr>
<tr>
<td>relation extraction</td>
<td>79, 582</td>
</tr>
<tr>
<td>Relation extraction</td>
<td>604</td>
</tr>
<tr>
<td>Requirements Engineering</td>
<td>297</td>
</tr>
<tr>
<td>residual network</td>
<td>528</td>
</tr>
<tr>
<td>Reuse</td>
<td>37</td>
</tr>
<tr>
<td>reverse-engineering</td>
<td>516</td>
</tr>
<tr>
<td>Risk</td>
<td>297</td>
</tr>
<tr>
<td>Risk Analysis</td>
<td>297</td>
</tr>
<tr>
<td>Risk Assessment</td>
<td>217</td>
</tr>
<tr>
<td>RNN</td>
<td>451</td>
</tr>
<tr>
<td>Term</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Root cause analysis</td>
<td>86</td>
</tr>
<tr>
<td>ROS package searching</td>
<td>411</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>SAT</td>
<td>538</td>
</tr>
<tr>
<td>Scalability</td>
<td>634</td>
</tr>
<tr>
<td>Scale Prediction</td>
<td>447</td>
</tr>
<tr>
<td>SDP-VP</td>
<td>79</td>
</tr>
<tr>
<td>Search strategies</td>
<td>510</td>
</tr>
<tr>
<td>securedesign patterns</td>
<td>98</td>
</tr>
<tr>
<td>Security Protocol</td>
<td>108</td>
</tr>
<tr>
<td>Semi-formal notation</td>
<td>303</td>
</tr>
<tr>
<td>Sensitivity Analysis</td>
<td>576</td>
</tr>
<tr>
<td>Sentence-BERT</td>
<td>79</td>
</tr>
<tr>
<td>sequence matching</td>
<td>205</td>
</tr>
<tr>
<td>Sequence Modeling</td>
<td>399</td>
</tr>
<tr>
<td>service</td>
<td>15</td>
</tr>
<tr>
<td>Similarity</td>
<td>622</td>
</tr>
<tr>
<td>smart learning</td>
<td>451</td>
</tr>
<tr>
<td>Social Aspects</td>
<td>229</td>
</tr>
<tr>
<td>social network</td>
<td>487</td>
</tr>
<tr>
<td>Software Architecture Evaluation</td>
<td>45</td>
</tr>
<tr>
<td>Software Engineering</td>
<td>229</td>
</tr>
<tr>
<td>Software maintenance and documentation</td>
<td>241</td>
</tr>
<tr>
<td>software quality</td>
<td>63</td>
</tr>
<tr>
<td>software requirement</td>
<td>285</td>
</tr>
<tr>
<td>Software Testing</td>
<td>554</td>
</tr>
<tr>
<td>Softwaremaintenance</td>
<td>255</td>
</tr>
<tr>
<td>SoftwareProjects</td>
<td>217</td>
</tr>
<tr>
<td>solving</td>
<td>628</td>
</tr>
<tr>
<td>SortingAlgorithm</td>
<td>25</td>
</tr>
<tr>
<td>Spatial-temporal forecast</td>
<td>309</td>
</tr>
<tr>
<td>Strand Space</td>
<td>108</td>
</tr>
<tr>
<td>structural similarity</td>
<td>429</td>
</tr>
<tr>
<td>Structured Threat Information Expression</td>
<td>67</td>
</tr>
<tr>
<td>Student Behaviours</td>
<td>189</td>
</tr>
<tr>
<td>Symbolic execution</td>
<td>538</td>
</tr>
<tr>
<td>T</td>
<td></td>
</tr>
<tr>
<td>technical capability evaluation model</td>
<td>381</td>
</tr>
<tr>
<td>temporal Graph Convolutional Network</td>
<td>417</td>
</tr>
<tr>
<td>Testing</td>
<td>166</td>
</tr>
<tr>
<td>testing AI software</td>
<td>560</td>
</tr>
<tr>
<td>Term</td>
<td>Page Numbers</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>text classification</td>
<td>7, 564</td>
</tr>
<tr>
<td>Text Classification</td>
<td>570, 576</td>
</tr>
<tr>
<td>The Reference Model (WRSPM Model)</td>
<td>297</td>
</tr>
<tr>
<td>Thematic Network</td>
<td>223</td>
</tr>
<tr>
<td>This approach in automatic</td>
<td>628</td>
</tr>
<tr>
<td>threat detection</td>
<td>67</td>
</tr>
<tr>
<td>threat intelligence</td>
<td>79</td>
</tr>
<tr>
<td>Throughput</td>
<td>634</td>
</tr>
<tr>
<td>Timed Automata</td>
<td>172</td>
</tr>
<tr>
<td>Tomek-link</td>
<td>75</td>
</tr>
<tr>
<td>tool that generates new</td>
<td>636</td>
</tr>
<tr>
<td>trace replay</td>
<td>205</td>
</tr>
<tr>
<td>Transformation</td>
<td>303</td>
</tr>
<tr>
<td>Transformer</td>
<td>321</td>
</tr>
<tr>
<td>Triangle Counting</td>
<td>387</td>
</tr>
<tr>
<td>Twitter</td>
<td>598</td>
</tr>
<tr>
<td>U</td>
<td></td>
</tr>
<tr>
<td>UAV</td>
<td>261</td>
</tr>
<tr>
<td>UML Diagrams</td>
<td>57</td>
</tr>
<tr>
<td>unified software development process</td>
<td>199</td>
</tr>
<tr>
<td>UPPAAL</td>
<td>150</td>
</tr>
<tr>
<td>Usability</td>
<td>275, 281</td>
</tr>
<tr>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Variational inference</td>
<td>309</td>
</tr>
<tr>
<td>Verification</td>
<td>102, 154</td>
</tr>
<tr>
<td>Verification Framework</td>
<td>160</td>
</tr>
<tr>
<td>W</td>
<td></td>
</tr>
<tr>
<td>wait-free algorithm</td>
<td>138</td>
</tr>
<tr>
<td>WebNLG</td>
<td>352</td>
</tr>
<tr>
<td>word embedding</td>
<td>285</td>
</tr>
<tr>
<td>Word Embedding</td>
<td>622</td>
</tr>
</tbody>
</table>
Proceedings of the 33rd International Conference on Software Engineering and Knowledge Engineering